Engenharia de Software I

Desenvolvimento ágil

Márcio Daniel Puntel

marcio@puntel.org

desenvolvimento de sistemas

- Segundo Friedrich Ludwig Bauer, "Engenharia de Software é a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe eficientemente em máquinas reais“
- ...salvação na **crise do software**.

- Teoria de construir prédios....
- NÃO é igual a desenvolver sistemas.

desenvolvimento de sistemas

- Engenharia de software “tentou” se ajustar
- Novas formas de desenvolver
- Objetos
- Componentes
- Frameworks

- ....e continuamos fazendo do mesmo tudo do mesmo jeito.

desenvolvimento ágil

Mudança de filosofia:

- Desenvolver sistemas não é dar uma de namorado malandro
- Não é construir um prédio
  As coisas não podem ser, e não são, “fixas” (cliente sabe disso)
- Mudança de software é bom!

desenvolvimento ágil

- Nova filosofia
  - Pensar diferente
- Coragem
  - Fazer diferente
- Simplificar
  - Fazer o mesmo com menor custo
- Manifesto ágil
  - Mudança de paradigma

princípios

http://agilemanifesto.org/

Assinado por 17 gurus da área de software; Utah (EUA); fev. 2001.

"Estamos descobrindo novas formas de trabalhar, onde passamos a valorizar:"

1. Indivíduos e interações **mais que** processos e ferramentas
2. Software funcionando **mais que** documentação abrangente
3. A colaboração com o cliente **mais que** negociar contratos
4. Responder a mudanças **mais que** seguir um plano

princípios

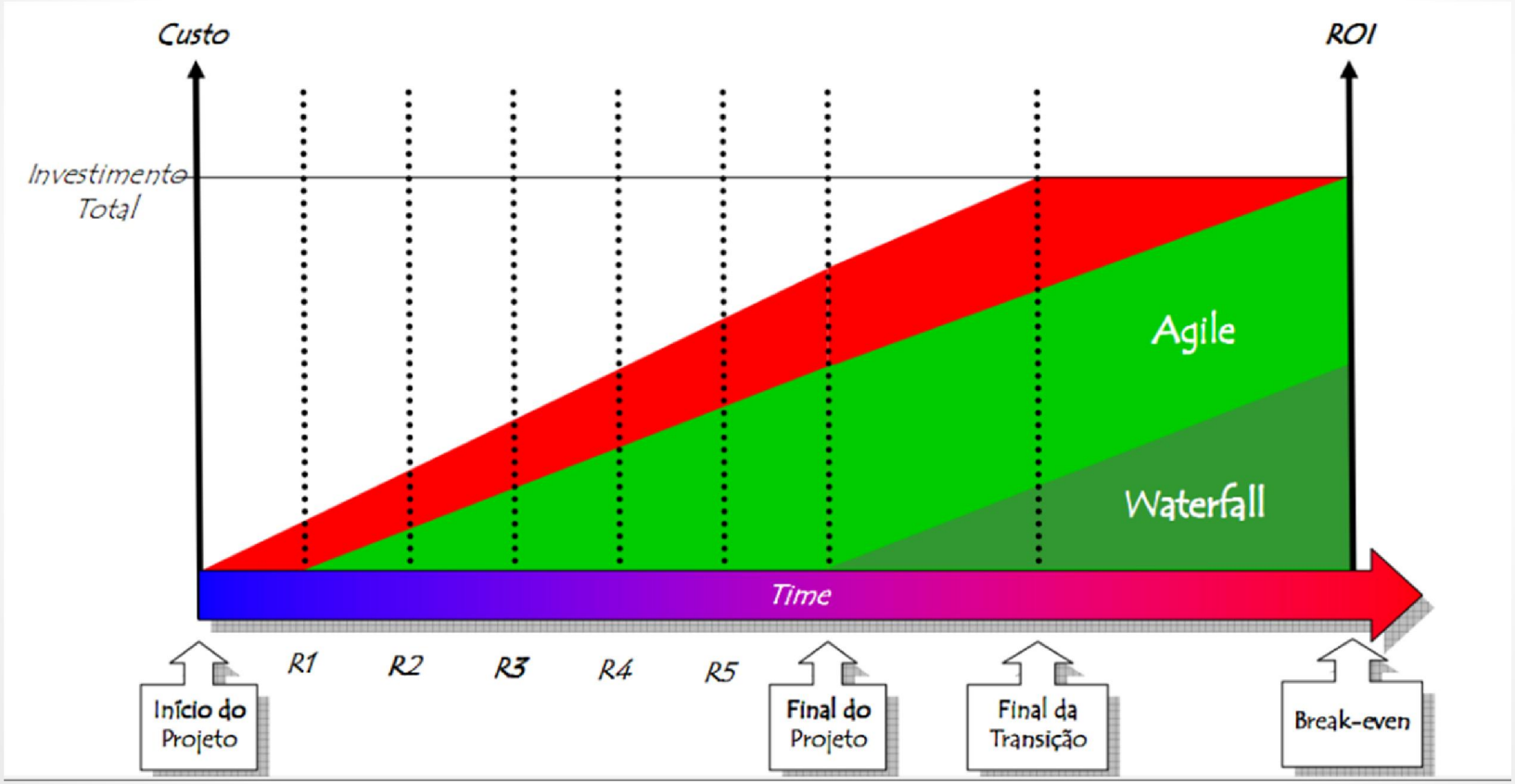
Baseado nos 12 princípios do Manifesto, veremos como usar, ou buscar embasamento, para engajar uma equipe com o projeto e seu cliente.

entregas frequentes

1. Você deve dar prioridade para satisfazer as **necessidade** do cliente com entregas **rápidas e frequentes** que agreguem valor ao negócio.

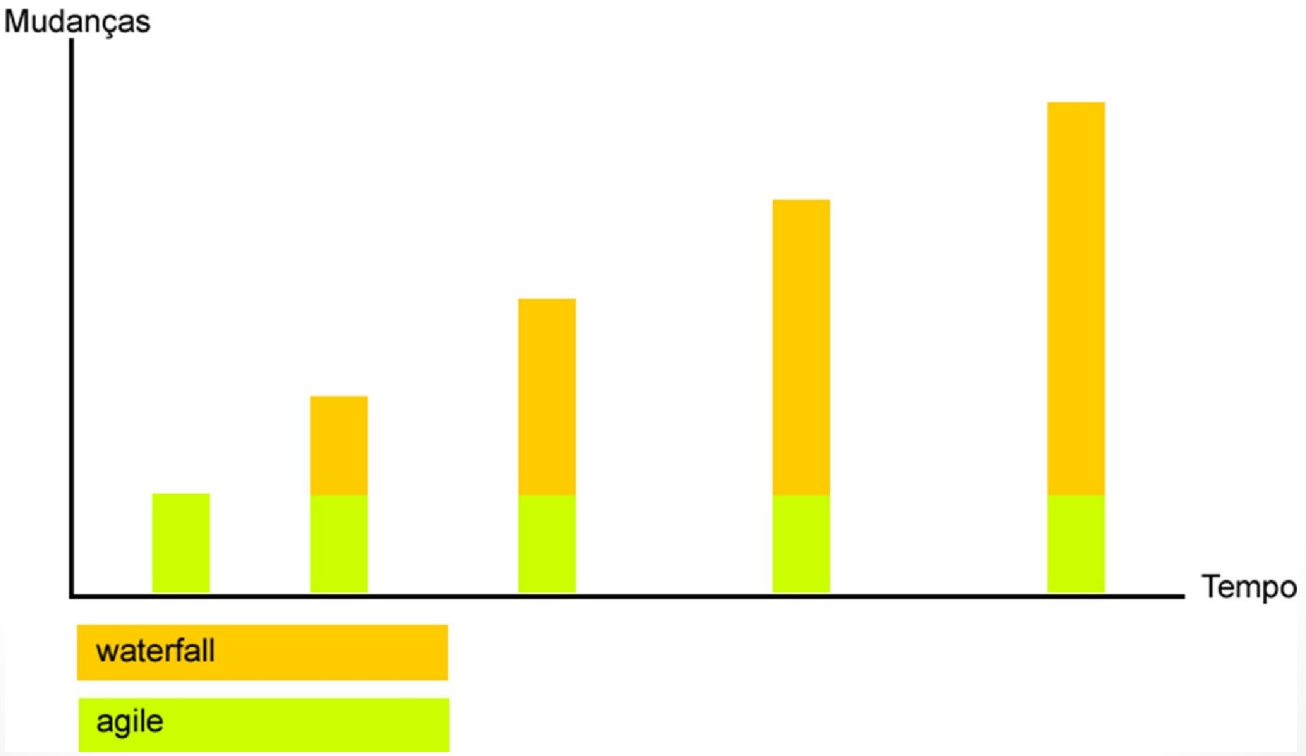
"Não me importa o quanto sua equipe pode ser ótima.....mas sim o quanto ela pode deixar meu cliente feliz."

entregas frequentes



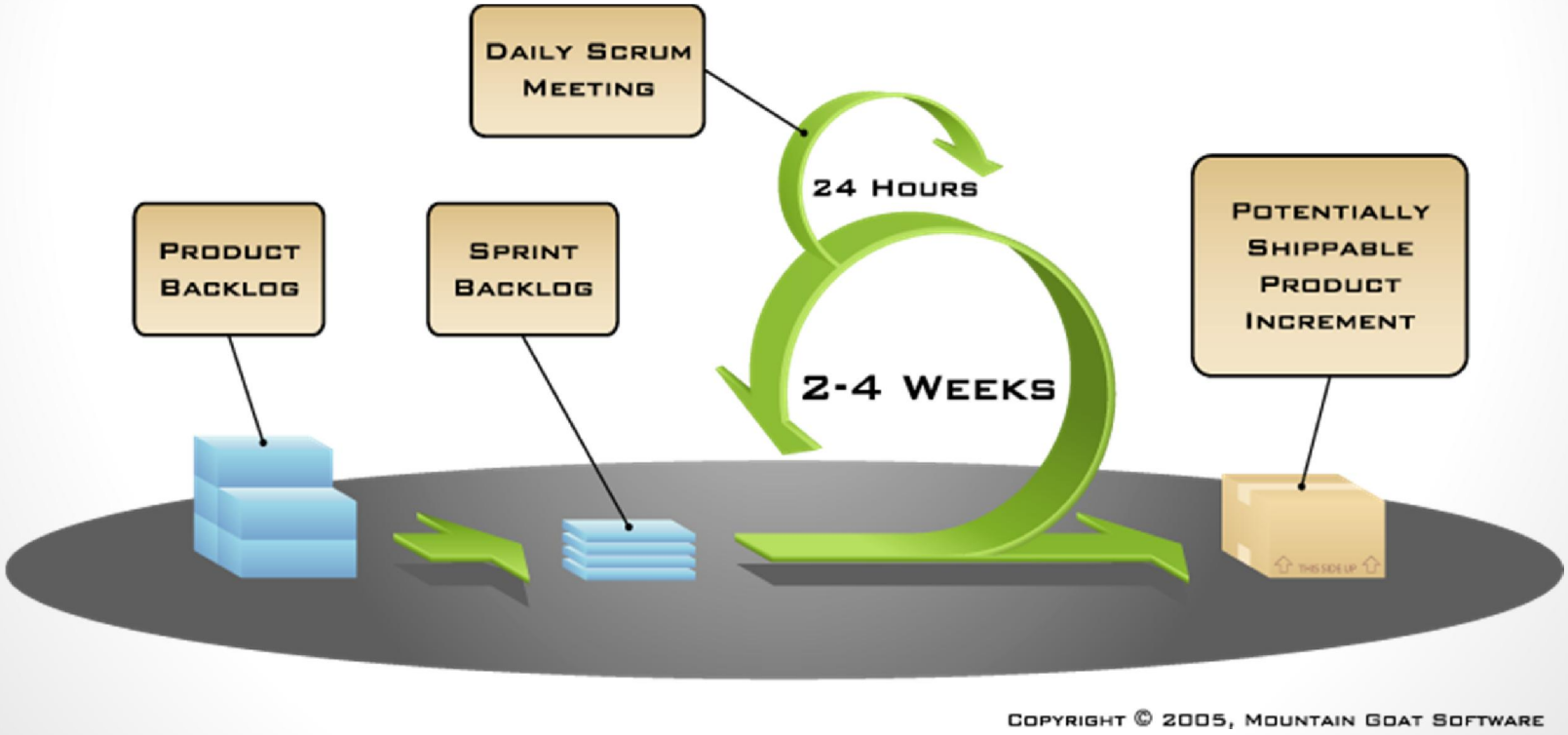
atender à mudanças

2. Você deve estar preparado para **atender à mudanças** durante **todo o projeto**.



entregar software funcionando

3. Você deve estar preparado para entregar **software funcionando** num **curto período** de tempo (iteração).



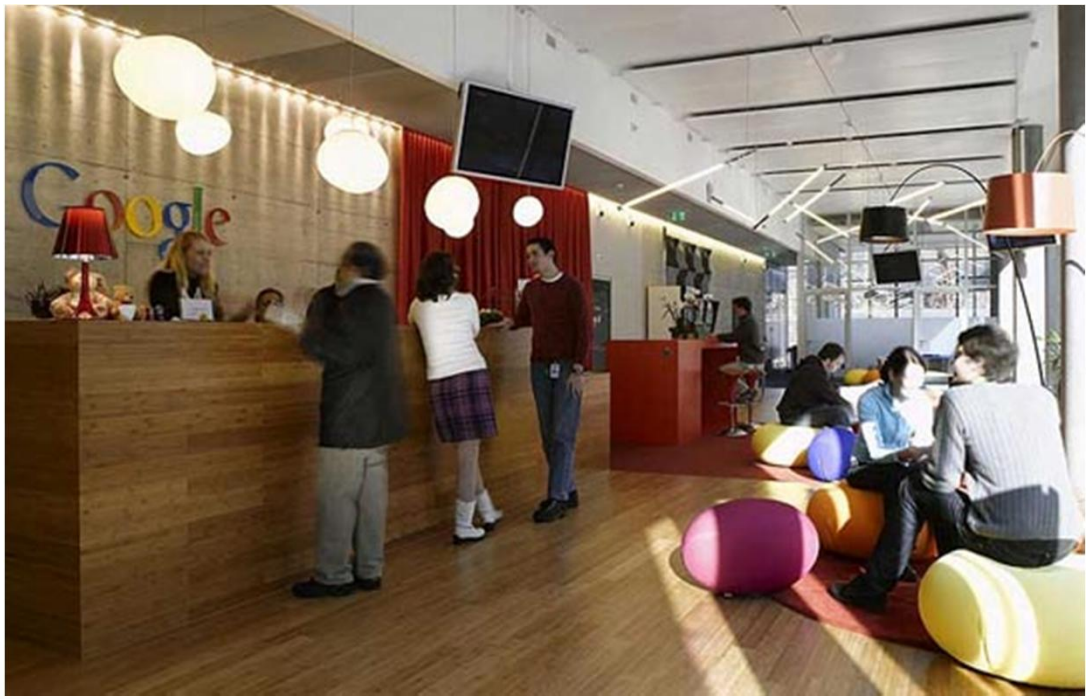
participação do cliente

4. Você deve estar preparado aceitar e incentivar a **participação do cliente** durante **todo o projeto**.

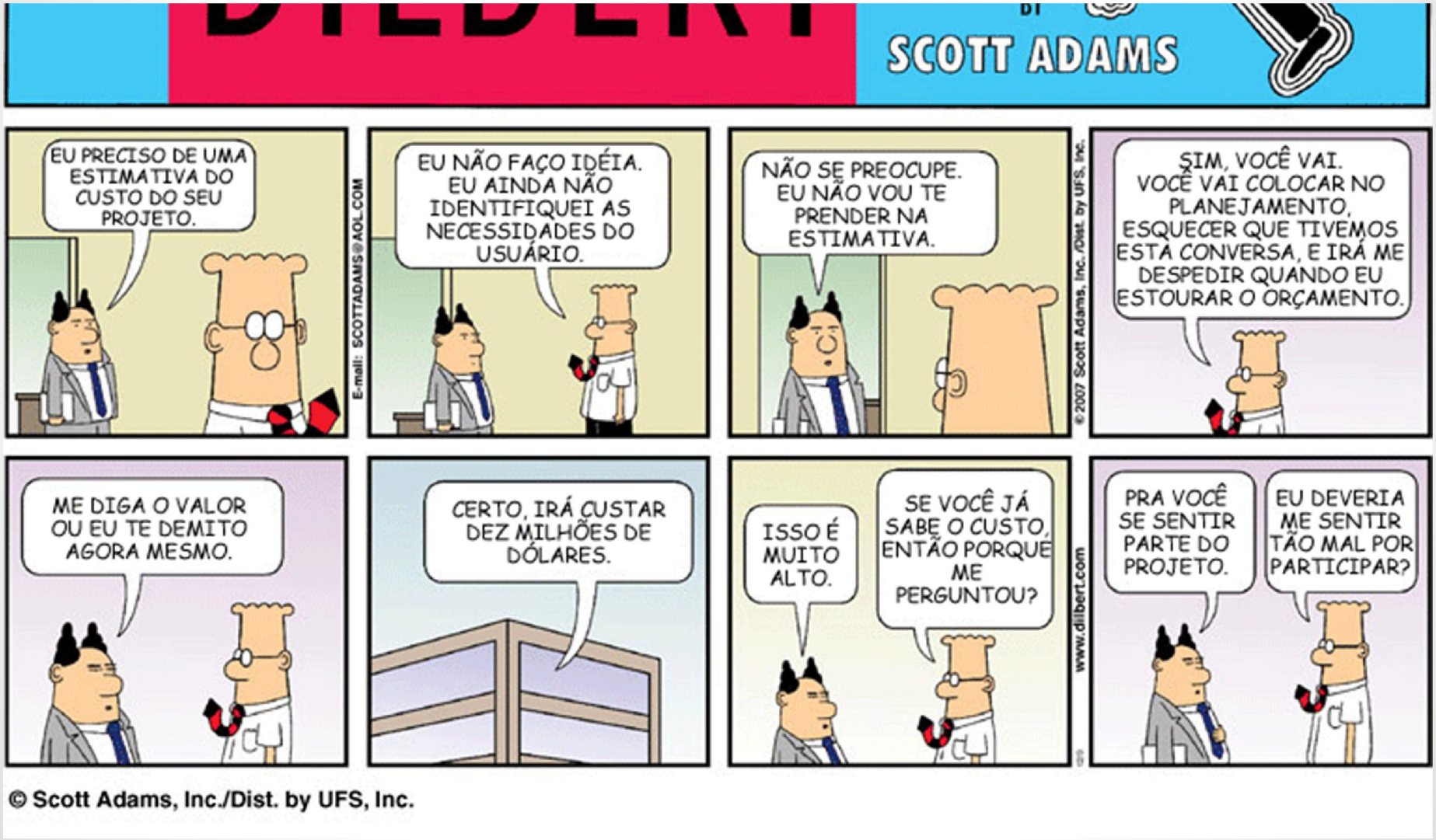
participação do cliente

ambiente vs. motivação

5. A**mbiente que dê motivação e suporte** para a equipe, assim como **confiança** no trabalho a ser realizado.



ambiente vs. motivação



comunicação com o cliente

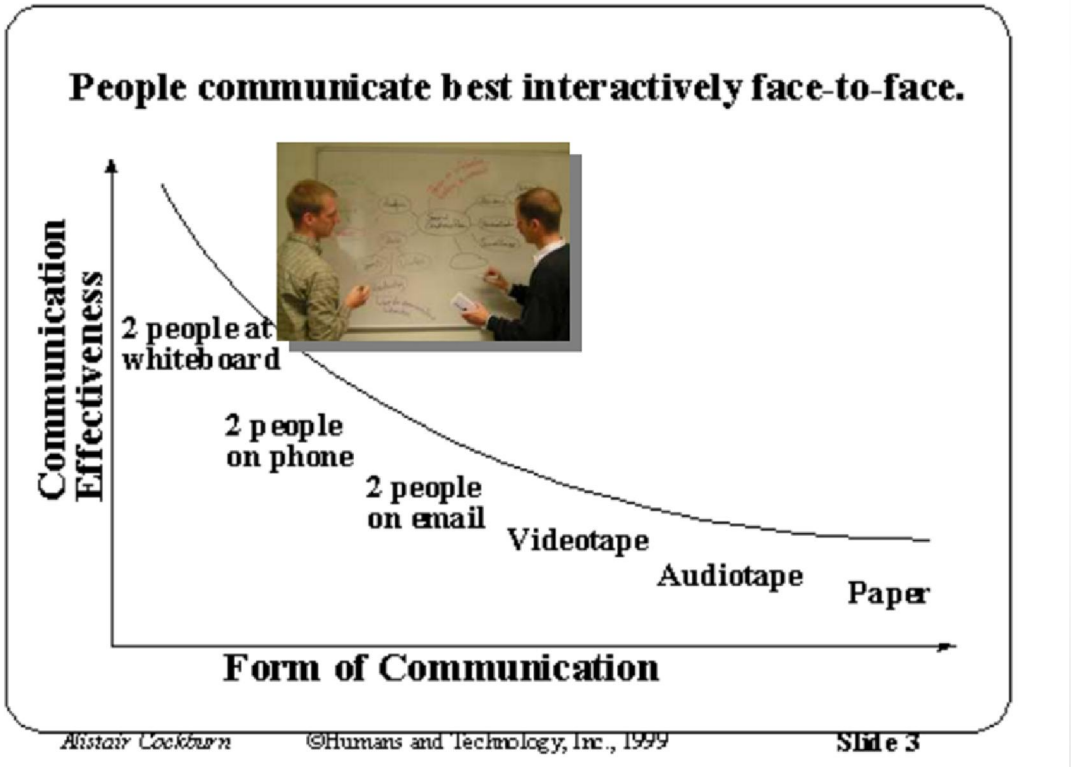
6. Buscar a **comunicação** com os clientes com diversas ferramentas, **preferencialmente presencial**.

Comunicação:

Palavras: 7%

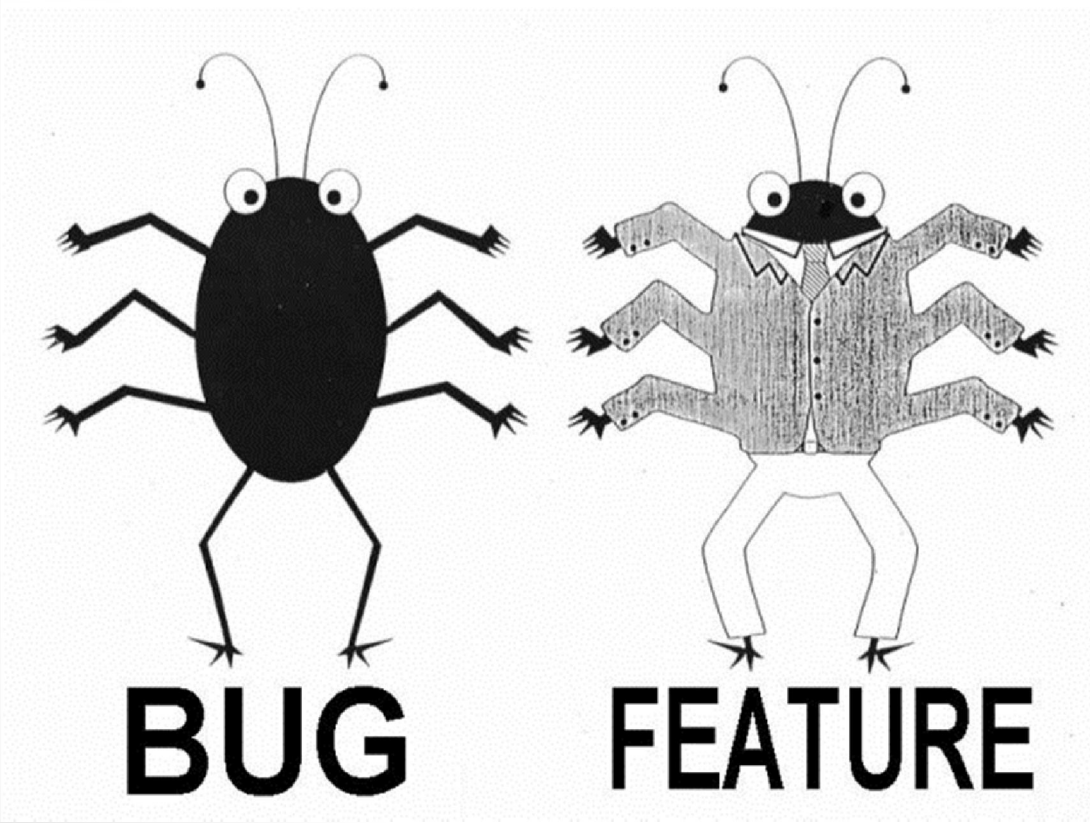
Sons: 38%

Fisiologia: 55%



medir com software funcionando

7. S**oftware funcionando** é a melhor medida de **progresso e desempenho**.



"Programadores com as mesmas qualificações, os mesmos recursos, o mesmo plano, as mesmas ferramentas, o mesmo local de trabalho e mesmos computadores **irão** desenvolver códigos diferentes."

medir com software funcionando

Para isso, investir em:

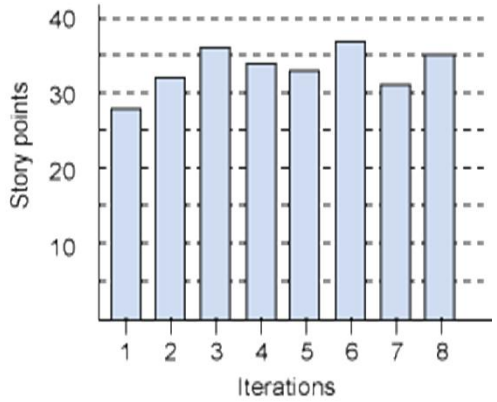
- Testes unitários;
- Testes funcionais;
- Automatização de testes;
- Integração.

medir com software funcionando

ritmo constante

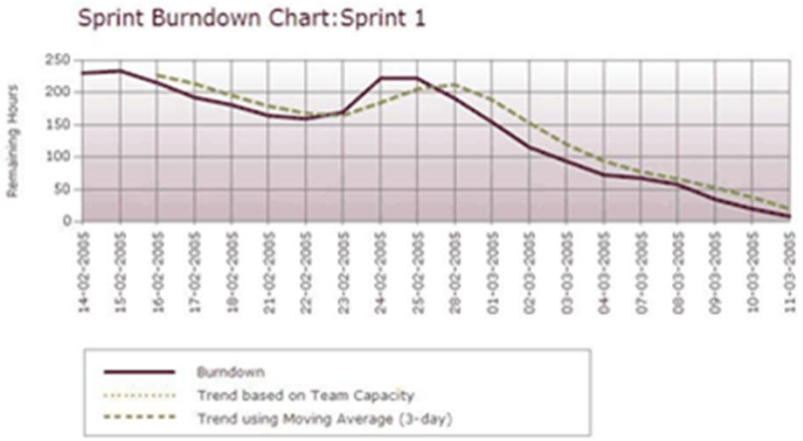
8. M**anter o ritmo** do projeto **constante** indefinidamente.

Ter controle do tempo das iterações.



ritmo constante

Ter controle da velocidade da equipe.



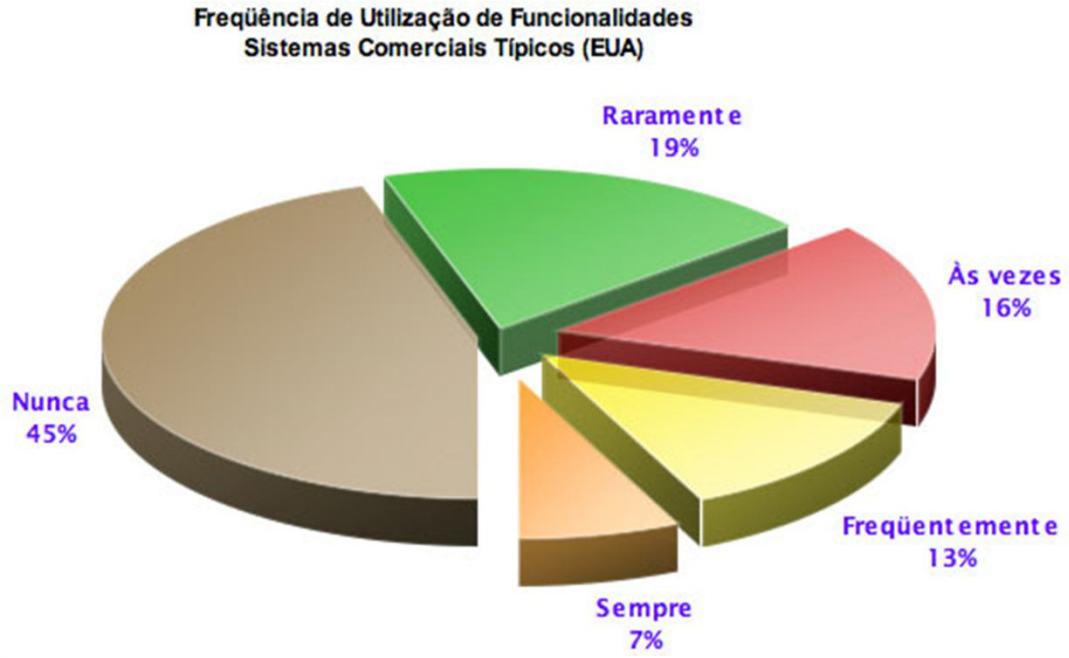
design também é importante

9. Dispensar atenção permanente ao **design** além da **excelência técnica**.

- Refactoring
- Patterns
- Visual também é importante
- Buscar alternativas (pair programming)

aumentar a redução de desperdício

10. M**aximizar** a **redução** do que tem que ser feito.



equipes auto-organizadas

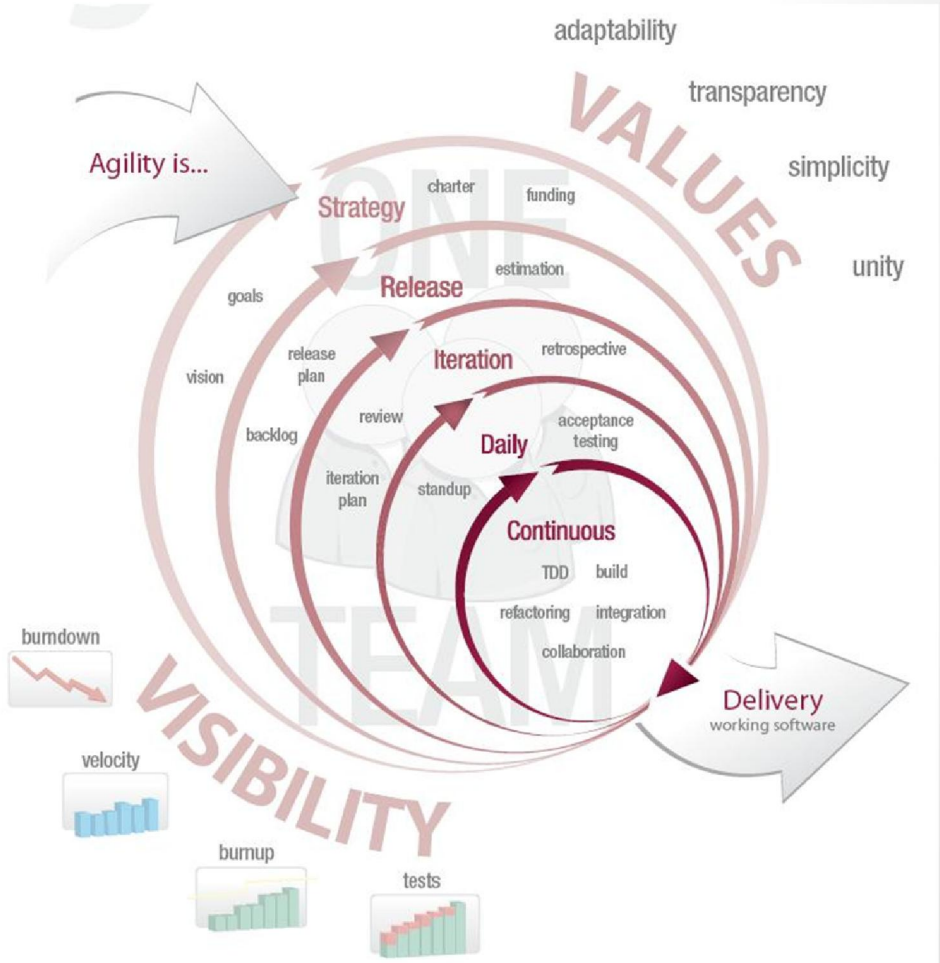
11. Equipe **auto-organizada** suficientemente para criar as melhores soluções para o projeto.

- Papéis bem definidos
- Bons líderes
- Comprometimento
- Profissionalismo

aprendizagem constante

12. Equipe seria capaz de, **regularmente**, refletir como **aprimorar as técnicas** usadas.

- ✓ Reuniões de retrospectiva
- ✓ Feedback do cliente
- ✓ Rever Medidores
- ✓ Reavaliar Métricas



ciclo básico ágil

- ✓Programa tem projetos pequenos;
- ✓Projetos tem iterações;
- ✓Interações tem um build validado;
- ✓Build validado tem build diário;
- ✓Build diário tem tarefas.

10 boas razões para usar

1. Melhoria contínua e rápida
2. Feedback constante do cliente
3. Comunicação
4. Flexibilidade
5. Aceitação de mudanças
6. Integração da equipe
7. Refactoring
8. Testes
9. Documentação eficaz
10. Produto esperado = Satisfação do cliente

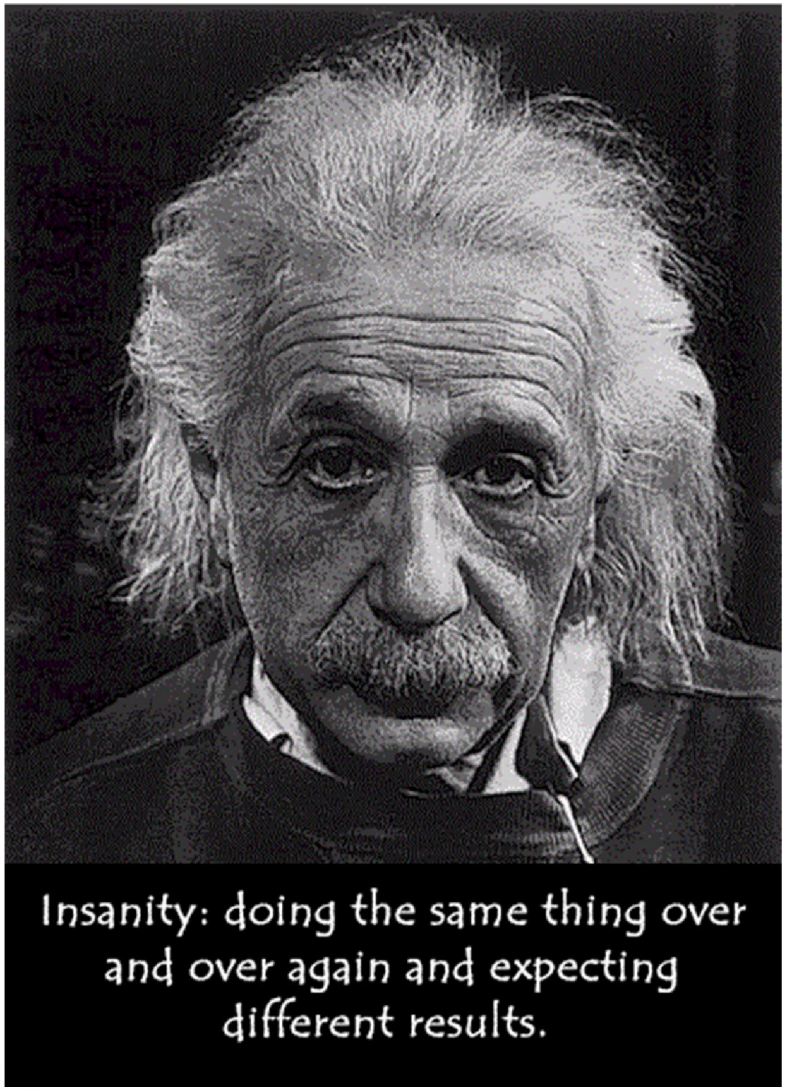
problemas

- Pessoas fazem software!
- Colegas são pessoas...que nunca entendem
- Chefes são pessoas...que são gargalos
- Clientes são pessoas...que não sabem o que querem
- Programadores são pessoas...que fazem lixo (desperdício)

...

- ✓ Logo, agilidade é, também:
- ✓ aprender a evitar os desperdícios;
- ✓ Troca de experiência
- ✓ Confiança
- ✓ Comunicação
- ✓ Equipe eficaz

pense nisso...


Insanity: doing the same thing over
and over again and expecting
different results.