

SURGIMENTOS DOS MÉTODOS ÁGEIS



Na década de 1980 e início da de 1990, havia uma visão generalizada de que a melhor maneira para conseguir o melhor software era por meio de um planejamento cuidadoso. O desenvolvimento de software era rigoroso e controlado. Essa percepção veio de profissionais responsáveis pelo desenvolvimento de sistemas de software grandes e duradouros, como sistemas aeroespaciais.

Essa abordagem em sistemas de pequeno e médio porte era ineficaz, pois gastava-se mais tempo em análises de como o sistema deve ser desenvolvido do que no desenvolvimento de programas e testes.

Na década de 1990, começaram a surgir métodos alternativos para desenvolvimento de sistemas. Diferente dos tradicionais, estes métodos não eram excessivamente regrados, lentos e burocráticos. Permitia que a equipe de desenvolvimento focasse no software em si, e não em sua concepção e documentação.

Em 2001 um grupo de 17 especialistas se reuniu em Utah nos Estados Unidos para debater a melhor forma de desenvolver softwares de uma maneira mais simples, rápida e centralizada em pessoas. Eles cunharam o termo **Métodos Ágeis** e criaram o **Manifesto Ágil**.

O QUE SÃO MÉTODOS ÁGEIS

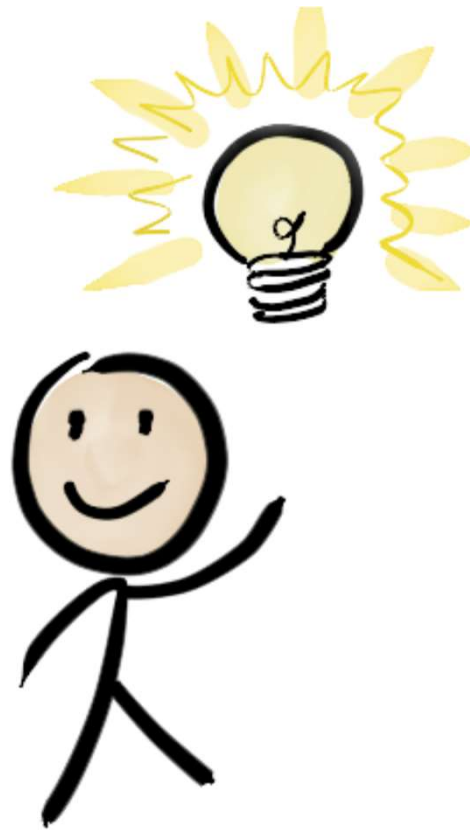
- Os métodos ágeis são métodos de desenvolvimento incremental.
- O software não é desenvolvido como uma única unidade, mas como uma série de incrementos. Cada incremento inclui uma nova funcionalidade do sistema.
- Os incrementos são pequenos e, normalmente as novas versões do sistema são criadas e disponibilizadas aos clientes.
- Destinam-se a entregar o software rapidamente aos clientes em funcionamento, e estes podem, em seguida, propor alterações e novos requisitos a serem incluídos.
- Eles são mais adequados ao desenvolvimento de aplicativos nos quais os requisitos mudam rapidamente durante o processo de desenvolvimento.
- Têm como objetivo reduzir a burocracia do processo (documentação).

Entrega antecipada do **valor** comercial
com **menos burocracia**

valer

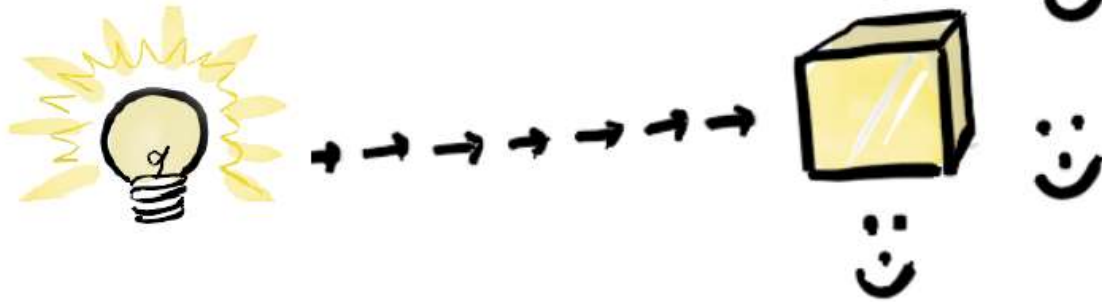
Alistair Cockburn

Todo produto/funcionalidade começa com uma **grande idéia!**

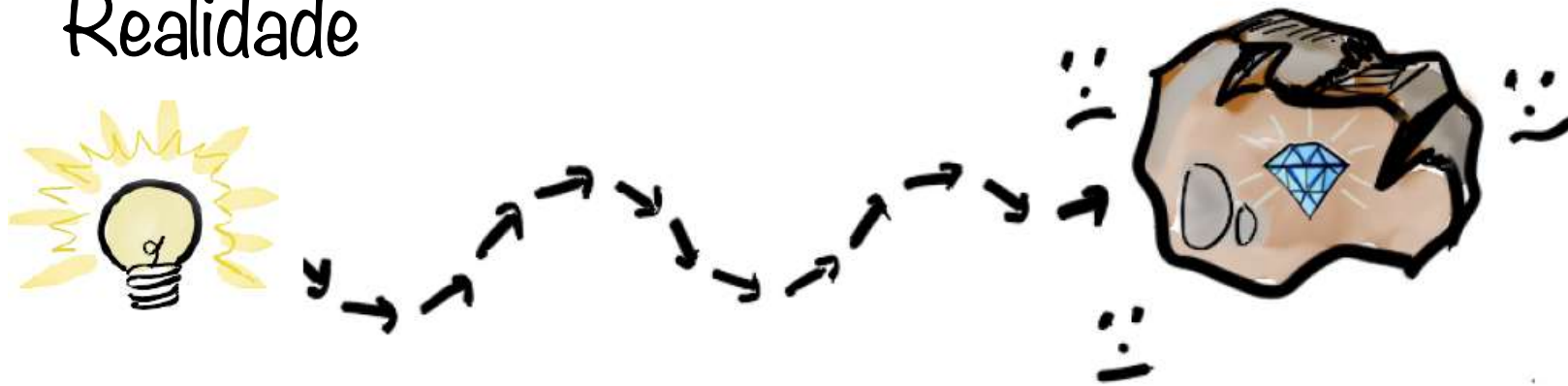


Infelizmente ... é provável que falhe

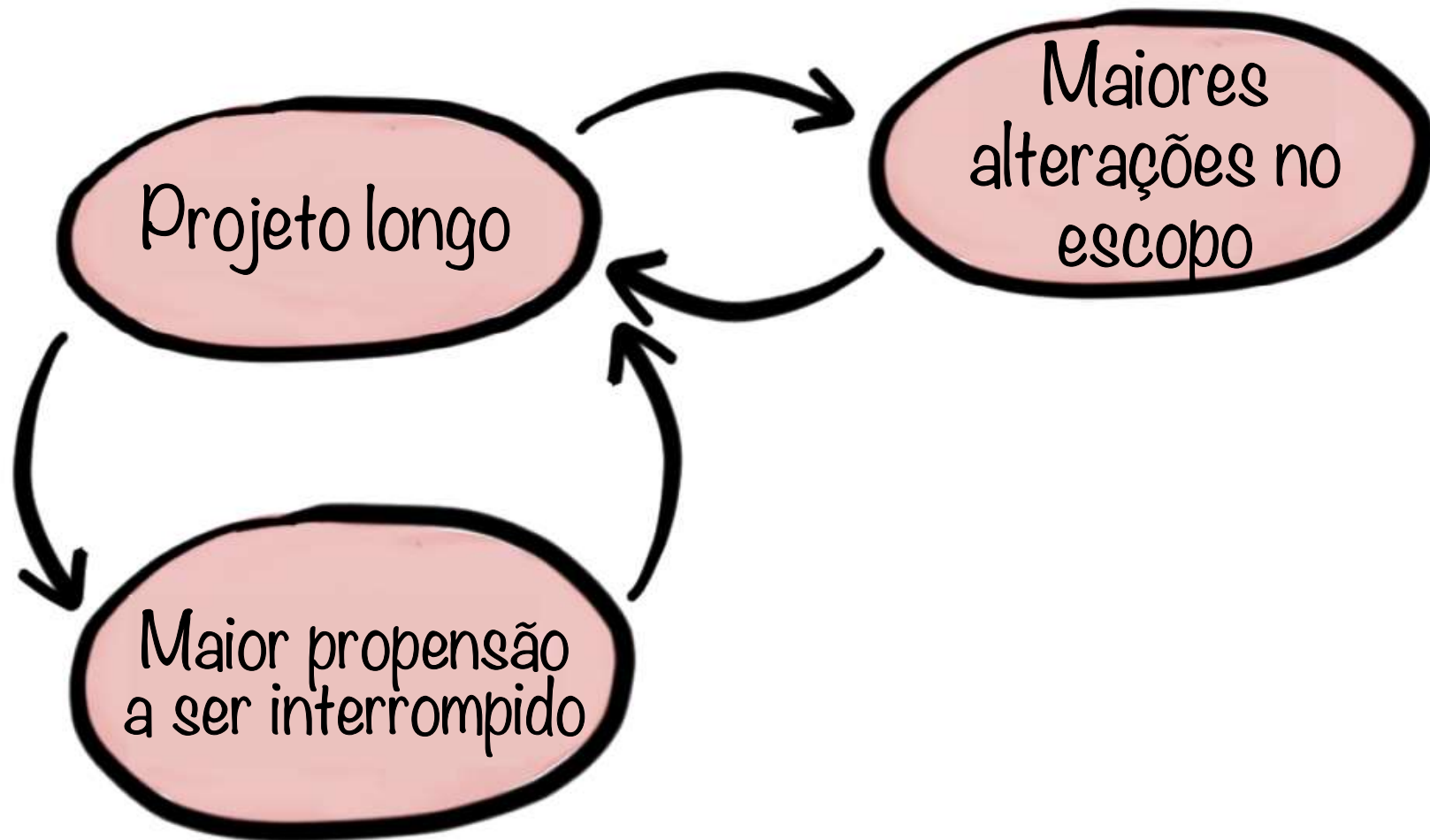
Projeto



Realidade



Projetos longos ficam mais longos

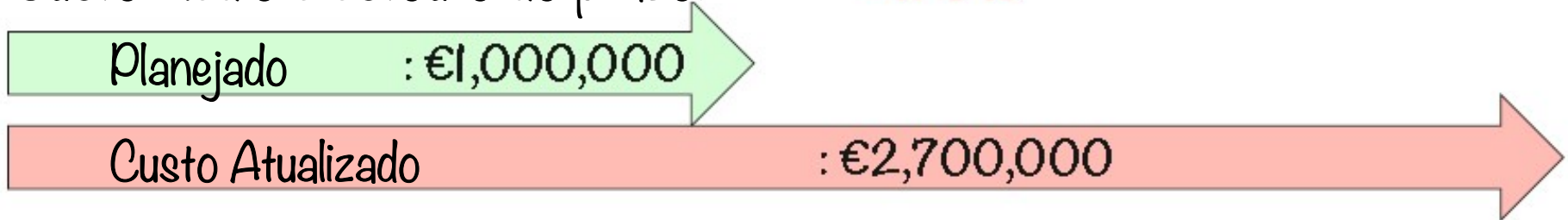


A maioria dos projetos de TI falham. E/ou atrasam.

O Standish Group estudou mais de 40,000 projetos em 10 anos

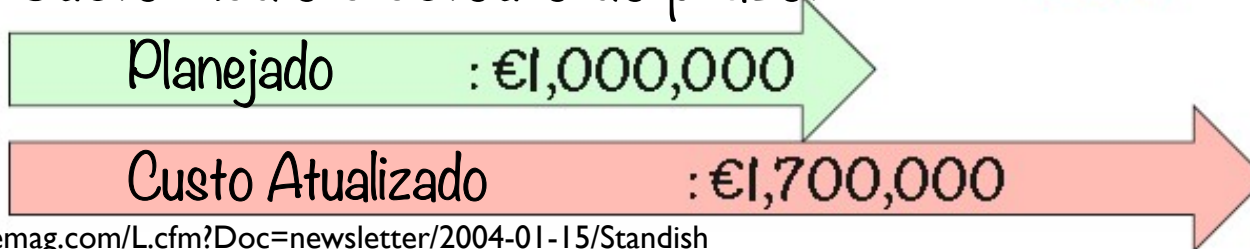
Taxa de sucesso de projetos de TI em 1994: **15%**

Custo médio e estouro do prazo: **≈170%**



Taxa de sucesso de projetos de TI em 2004: **34%**

Custo médio e estouro do prazo: **≈70%**



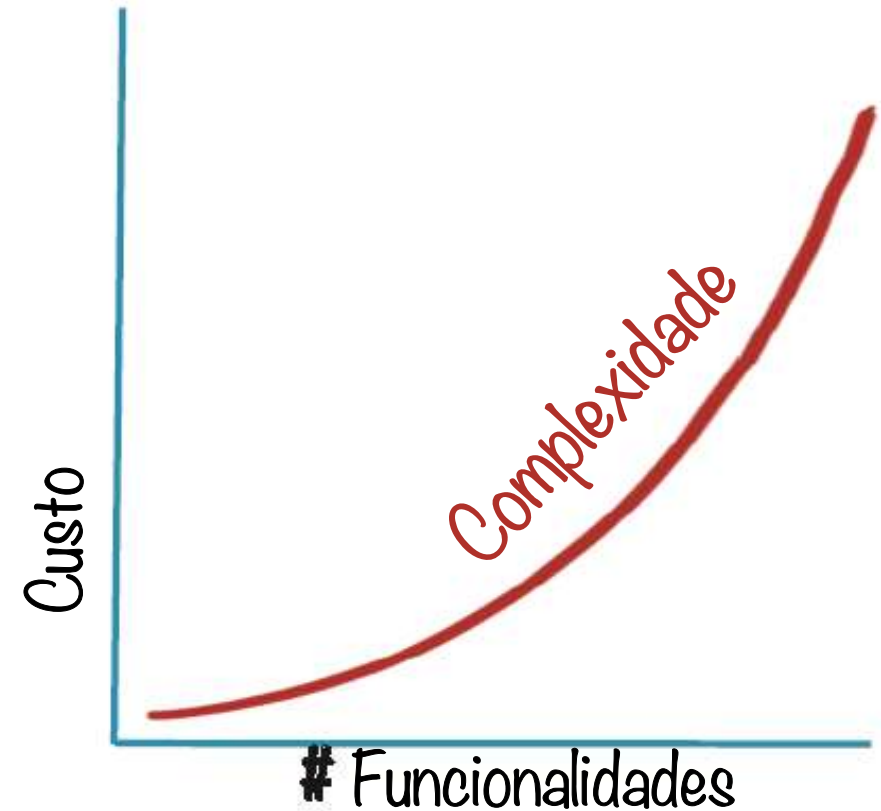
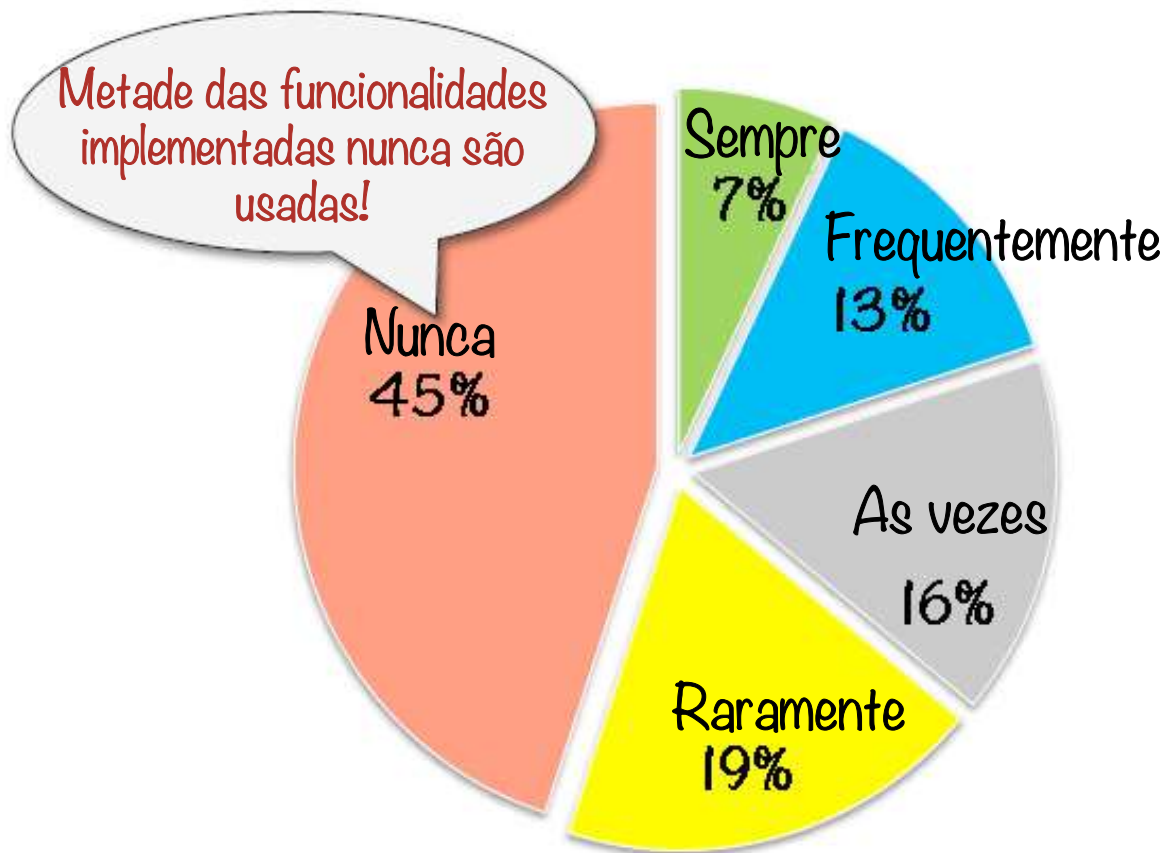
Fontes:

<http://www.softwaremag.com/L.cfm?Doc=newsletter/2004-01-15/Standish>

<http://www.infoq.com/articles/Interview-Johnson-Standish-CHAOS>

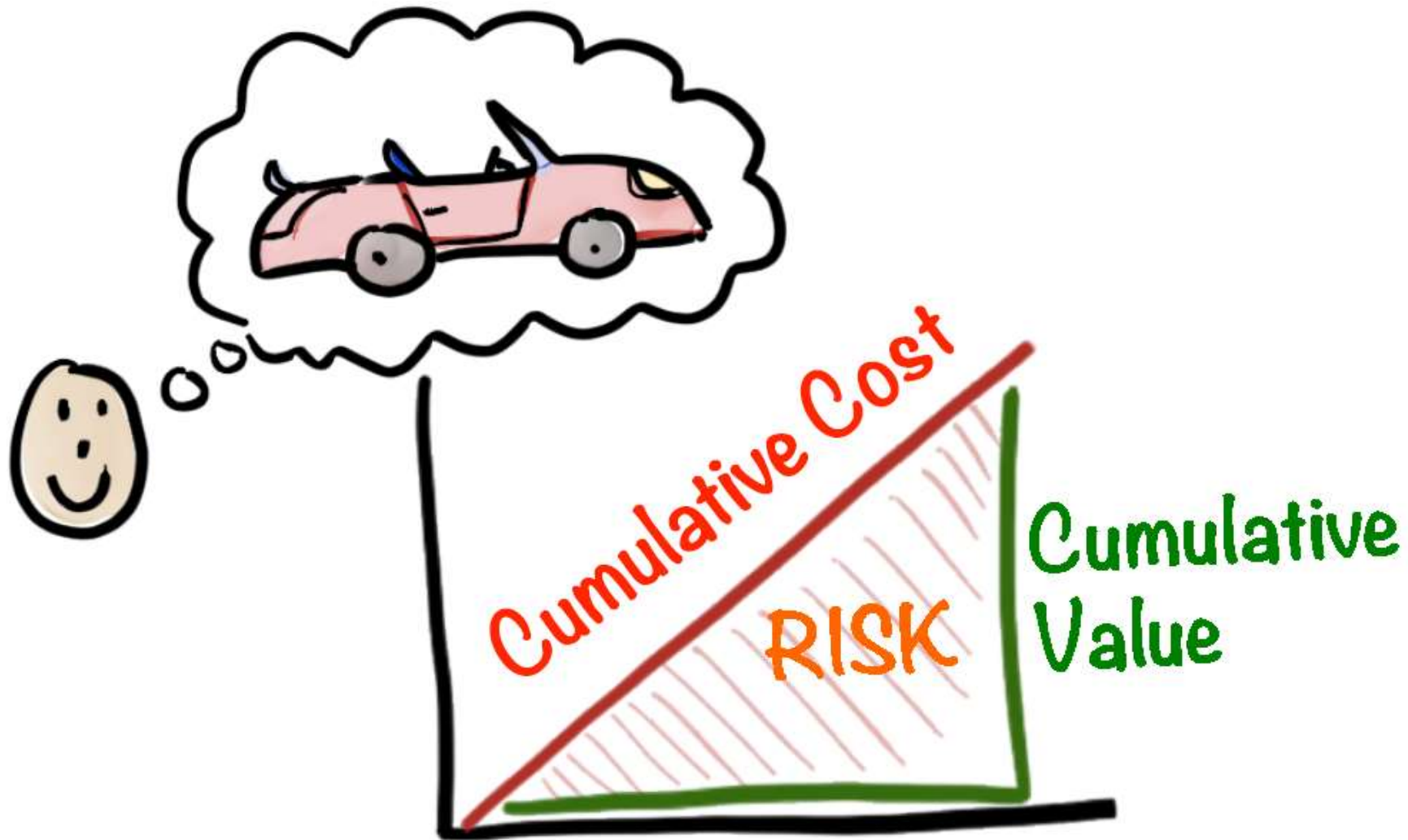
Temos tendência à implementar de forma errada

Recursos e funcionalidades usados em um sistema típico



Fonte:
Relatório do Standish Group divulgado no XP2002 por Jim Johnson

Grandes projetos = Grandes riscos



AGILE

Grandes projetos = Tiro de Canhão

Suposições:

- O cliente sabe o que ele quer
- Os desenvolvedores sabem como construir
- Nada vai mudar ao longo do caminho



Agile = míssil autodirecional

Suposições:

- O cliente descobre o que ele quer
- Os desenvolvedores descobrem como construí-lo
- As coisas mudam ao longo do caminho



METODOLOGIAS ÁGEIS

- Surgiram no início dos anos 2000 propondo nova abordagem de desenvolvimento.
- Reação às metodologias tradicionais com o intuito de criação de novas alternativas.
- Em 2001, 17 especialistas criaram a Aliança Ágil e através do Manifesto Ágil, popularizou-se o termo “Metodologia Ágil”.

CARACTERÍSTICAS FUNDAMENTAIS



flexibilidade



Modelo iterativo e incremental



Abordagem clássica vs. Abordagem ágil

	Clássica	Ágil
Desenvolvedor	hábil	ágil
Cliente	pouco envolvido	comprometido
Requisitos	conhecidos, estáveis	emergentes, mutáveis
Retrabalho	caro	barato
Planejamento	direciona resultados	resultados o direcionam
Foco	grandes projetos	Todos os tamanhos de projetos
Objetivo	controlar, em busca de alcançar o planejado	simplificar processo de desenvolvimento

1 - Não há especificação detalhada do sistema, e a documentação do projeto é minimizada ou gerada automaticamente pelo ambiente de programação. O documento de requisitos do usuário apenas define as características mais importantes do sistema.

2 - O sistema é desenvolvido em uma série de versões. Os usuários finais do sistema são envolvidos na especificação e avaliação de cada versão. Eles podem propor alterações e novos requisitos que devem ser implementados em uma versão posterior.

3 - Interfaces de usuário do sistema são feitas geralmente com um sistema iterativo de desenvolvimento que permite a criação rápida do projeto de interface por meio de desenho e posicionamento de ícones na interface.

O Manifesto Ágil é composto pela declaração de alguns valores e 12 princípios que fundamentam o desenvolvimento ágil de software.

Cada método ágil possui suas próprias práticas, entretanto todos em algum momento compartilham dos valores e princípios declarados no Manifesto Ágil.

MANIFESTO ÁGIL

Manifesto Ágil

<http://agilemanifesto.org/iso/ptbr/manifesto.html>

Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo.

Fev 11-13, 2001
Snowbird ski resort, Utah

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Manifesto Ágil

<http://agilemanifesto.org/iso/ptbr/manifesto.html>

Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:

Indivíduos e interações **mais** que processos e ferramentas
Software em funcionamento **mais** que documentação abrangente
Colaboração com o cliente **mais** que negociação de contratos
Responder a mudanças **mais** que seguir um plano

Ou seja, mesmo havendo valor nos itens à direita, valorizamos **mais** os itens à esquerda.

Autores do Manifesto Ágil:

Kent Beck - Uma das maiores referências do mundo ágil.

Jeff Sutherland e Ken Schwaber - Inventores do Scrum.

Martin Fowler - Importante referência em design para desenvolvedores.

Dave Thomas e Andrew Hunt - Pregam a simplicidade e leveza no desenvolvimento, além de metodologias centradas em pessoas.

Alistair Cockburn - Criador da família de métodos ágeis chamada de Crystal.

Ward Cunningham - Criador do método de design CRC e contribuiu para outras metodologias, incluindo XP.

Arie van Bennekum - Ativamente envolvido no consórcio DSDM (*Dynamic Systems Development Method*).

Brian Marick - Representante da comunidade de testes e das ideias do que o *Agile Testing* pode ser.

Jim Highsmith - Autor do método *Adaptive Software Development* (ASD) e do livro com mesmo nome.

Robert C. Martin - Experiente em XP.

Ron Jeffries - Primeiro coach em XP.

Jon Kern - Programador e arquiteto experiente em diversas linguagens.

Mike Beedle - Adotou Scrum e XP como metodologias ágeis com sucesso em diversos projetos.

Stephen J. Mellor - Foi coordenador do *Advisory Board* da revista IEEE Software por dez anos.

James Grenning - Um dos criadores da técnica conhecida como *Planning Poker*.

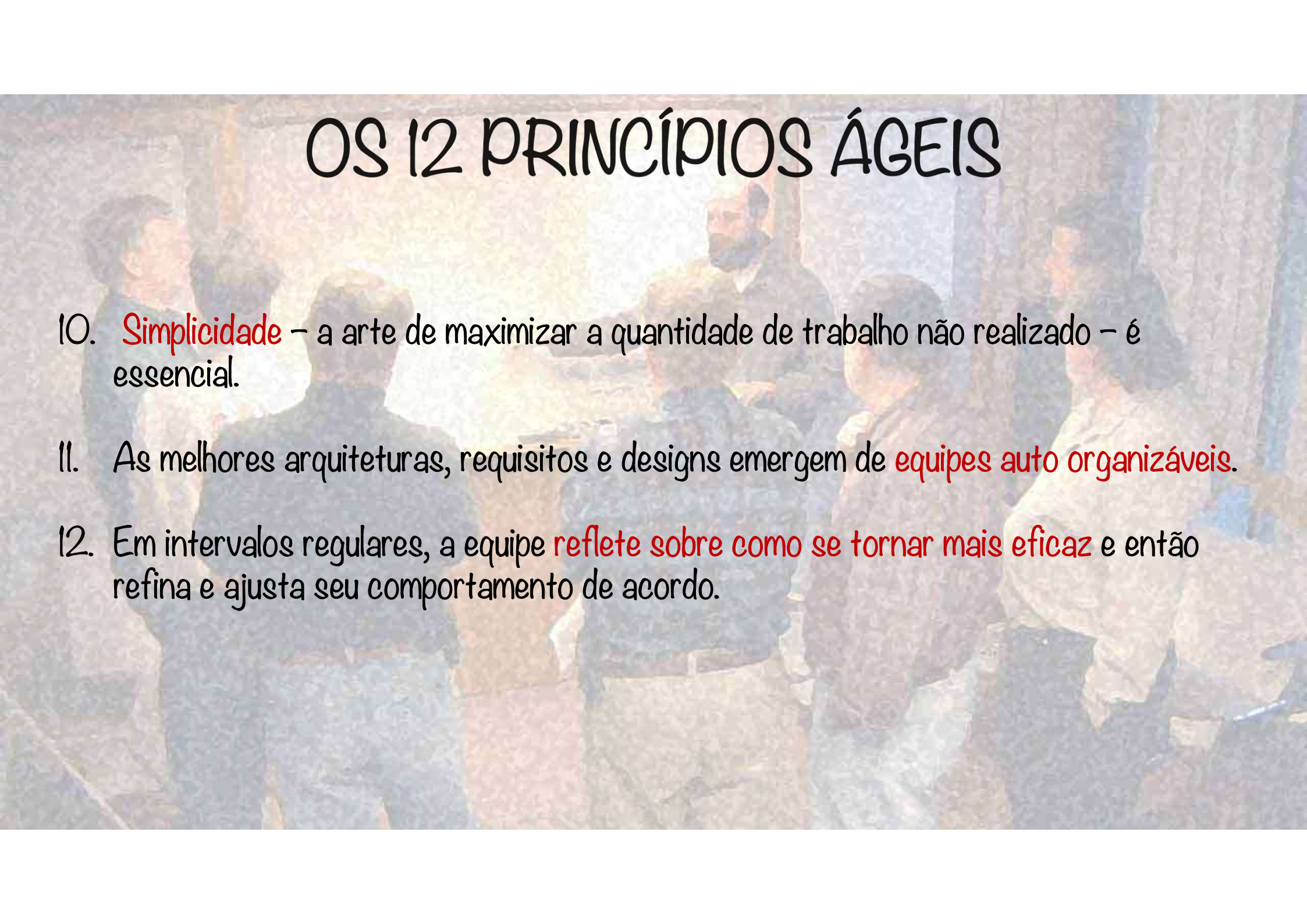
OS 12 PRINCÍPIOS ÁGEIS

1. Nossa maior prioridade é **satisfazer o cliente** através da entrega contínua e adiantada de software com valor agregado.
2. **Mudanças nos requisitos são bem-vindas**, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
3. **Entregar frequentemente software funcionando**, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
4. **Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto** por todo o projeto.

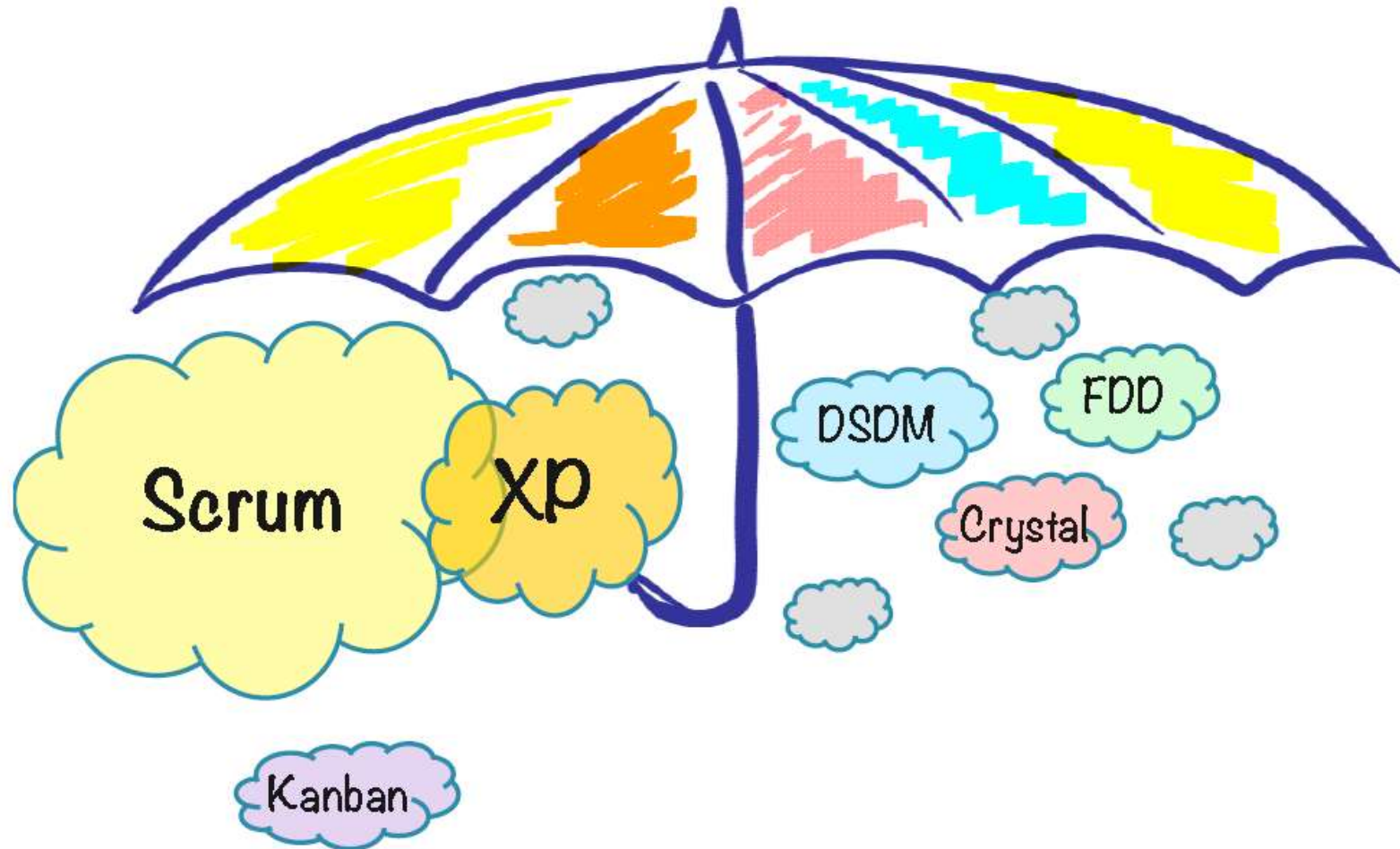
OS 12 PRINCÍPIOS ÁGEIS

5. Construa projetos em torno de **indivíduos motivados**. Dê a eles o ambiente e o suporte necessário e **confie** neles para fazer o trabalho.
6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de **conversa face a face**.
7. **Software funcionando** é a medida primária de progresso.
8. Os processos ágeis promovem **desenvolvimento sustentável**. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
9. Contínua atenção **à excelência técnica e bom design** aumenta a agilidade.

OS 12 PRINCÍPIOS ÁGEIS

- 
10. **Simplicidade** – a arte de maximizar a quantidade de trabalho não realizado – é essencial.
 11. As melhores arquiteturas, requisitos e designs emergem de **equipes auto organizáveis**.
 12. Em intervalos regulares, a equipe **reflete sobre como se tornar mais eficaz** e então refina e ajusta seu comportamento de acordo.

O “guarda chuva” Agile – uma família de métodos iterativos e incrementais

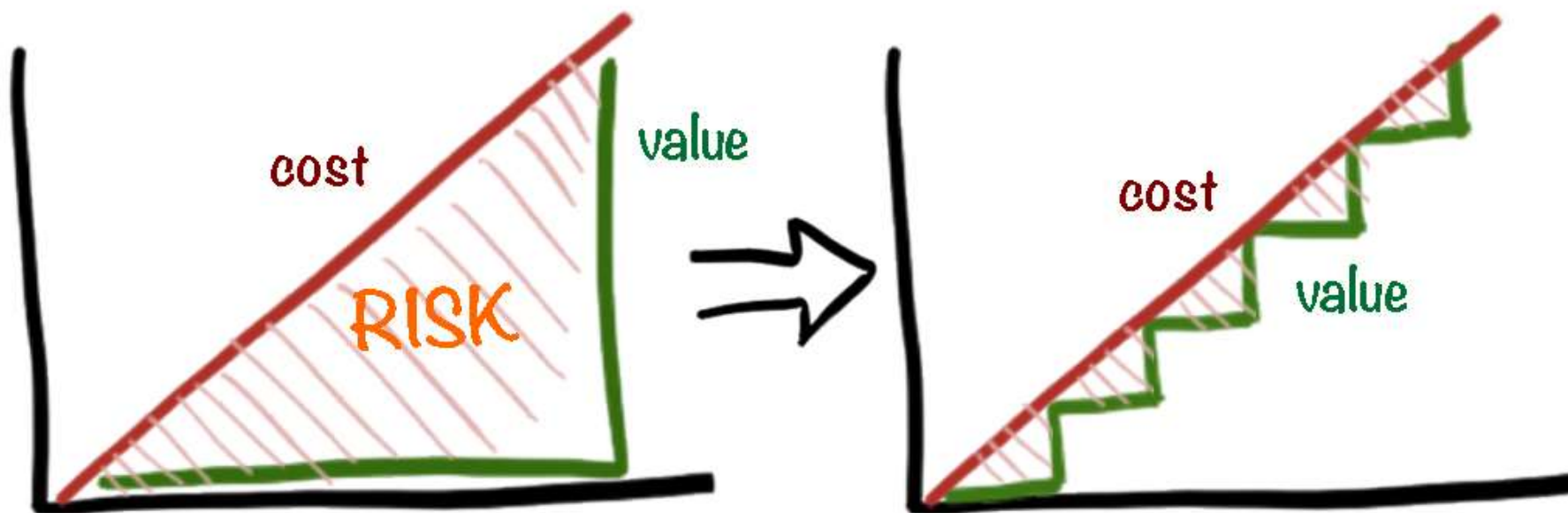
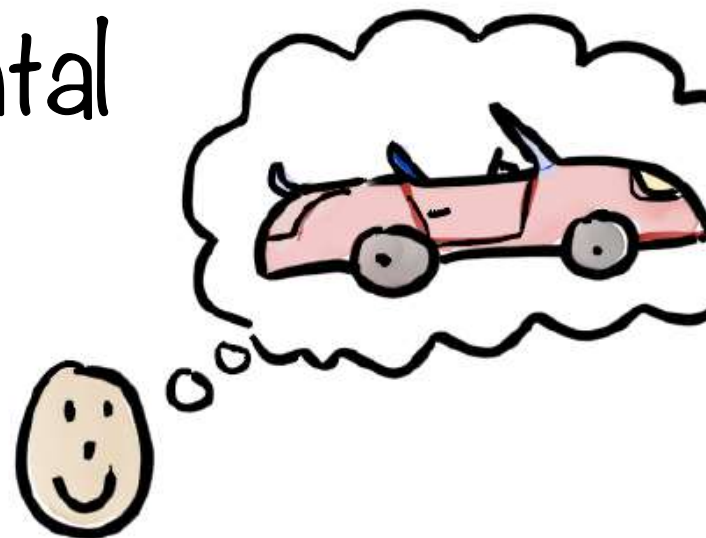


MUDANÇA DE PARADIGMA

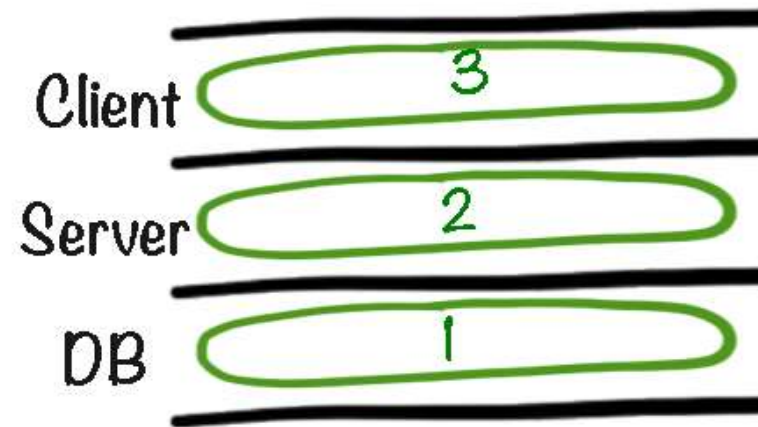
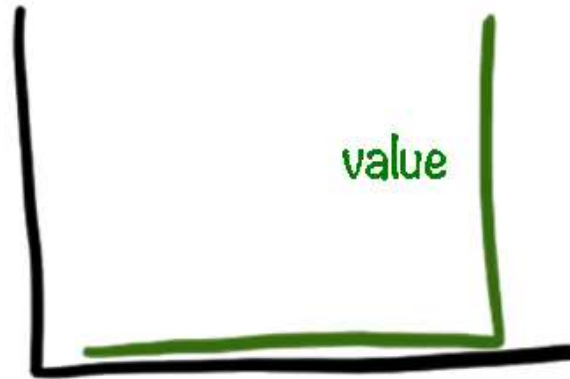
Agile = Iterativo + Incremental

↗
Não tente fazer tudo certo
desde o começo

↗
Não construa
tudo de uma vez



Incrementos não “Horizontais”



1



2

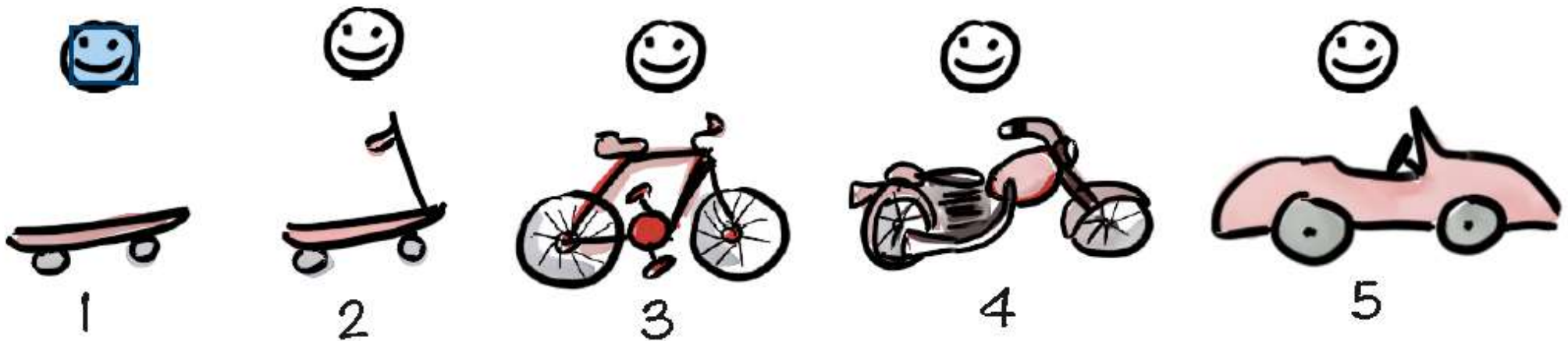
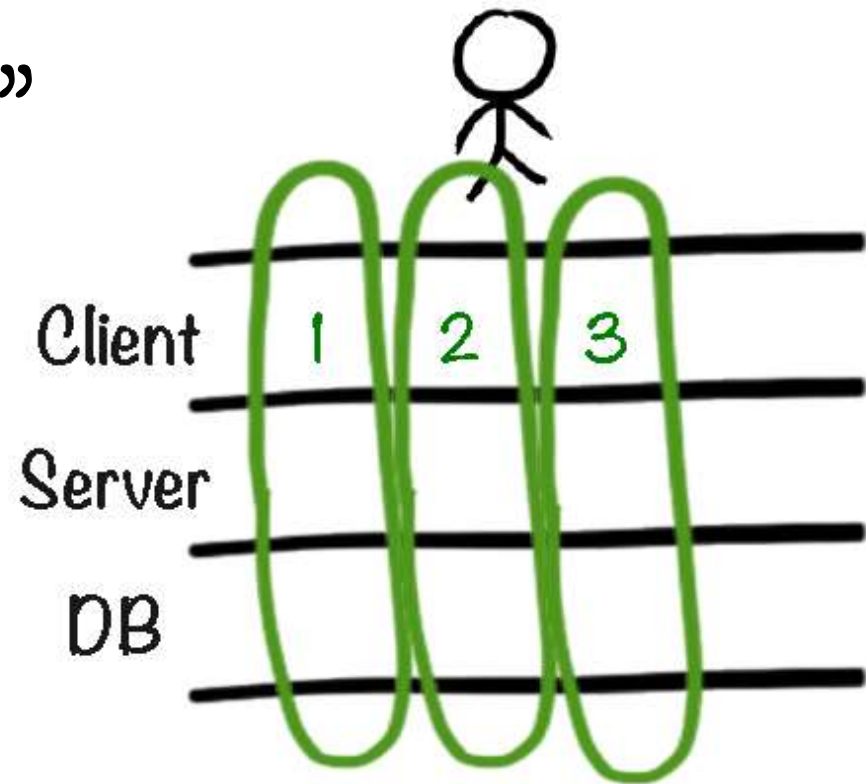
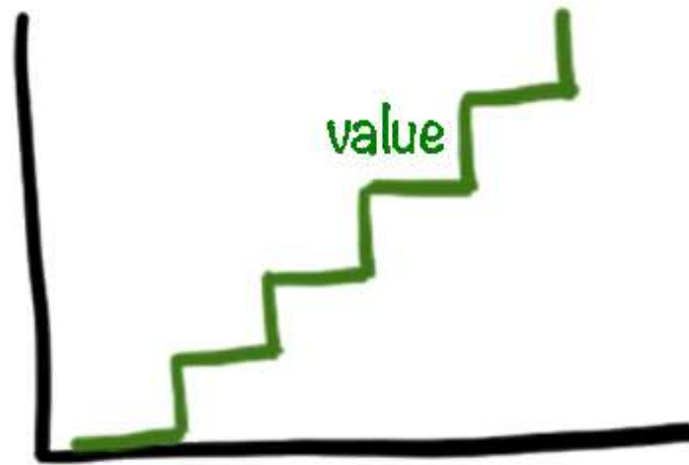


3



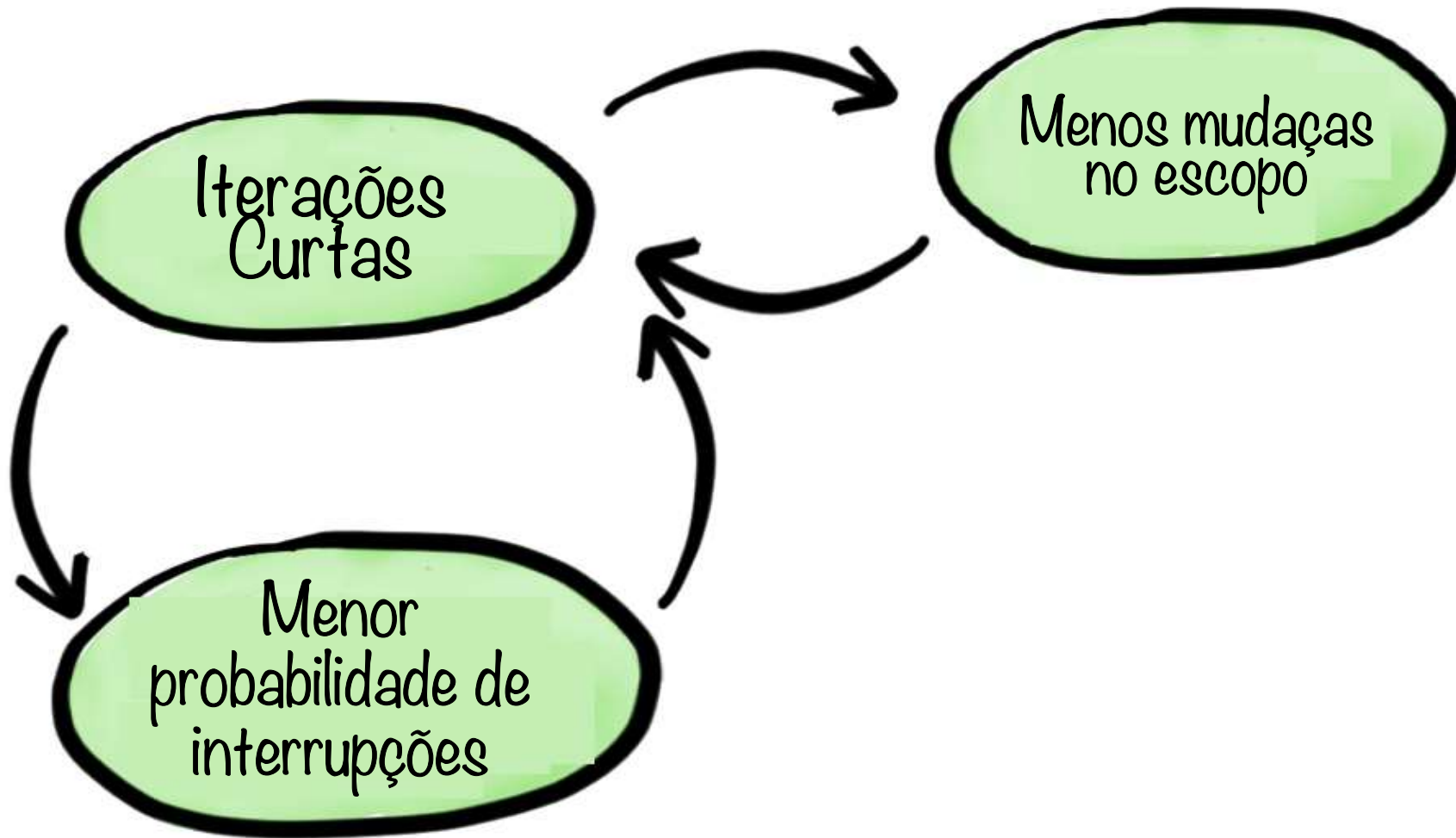
4

Incrementos “Verticais”



Mantenha as iterações curtas

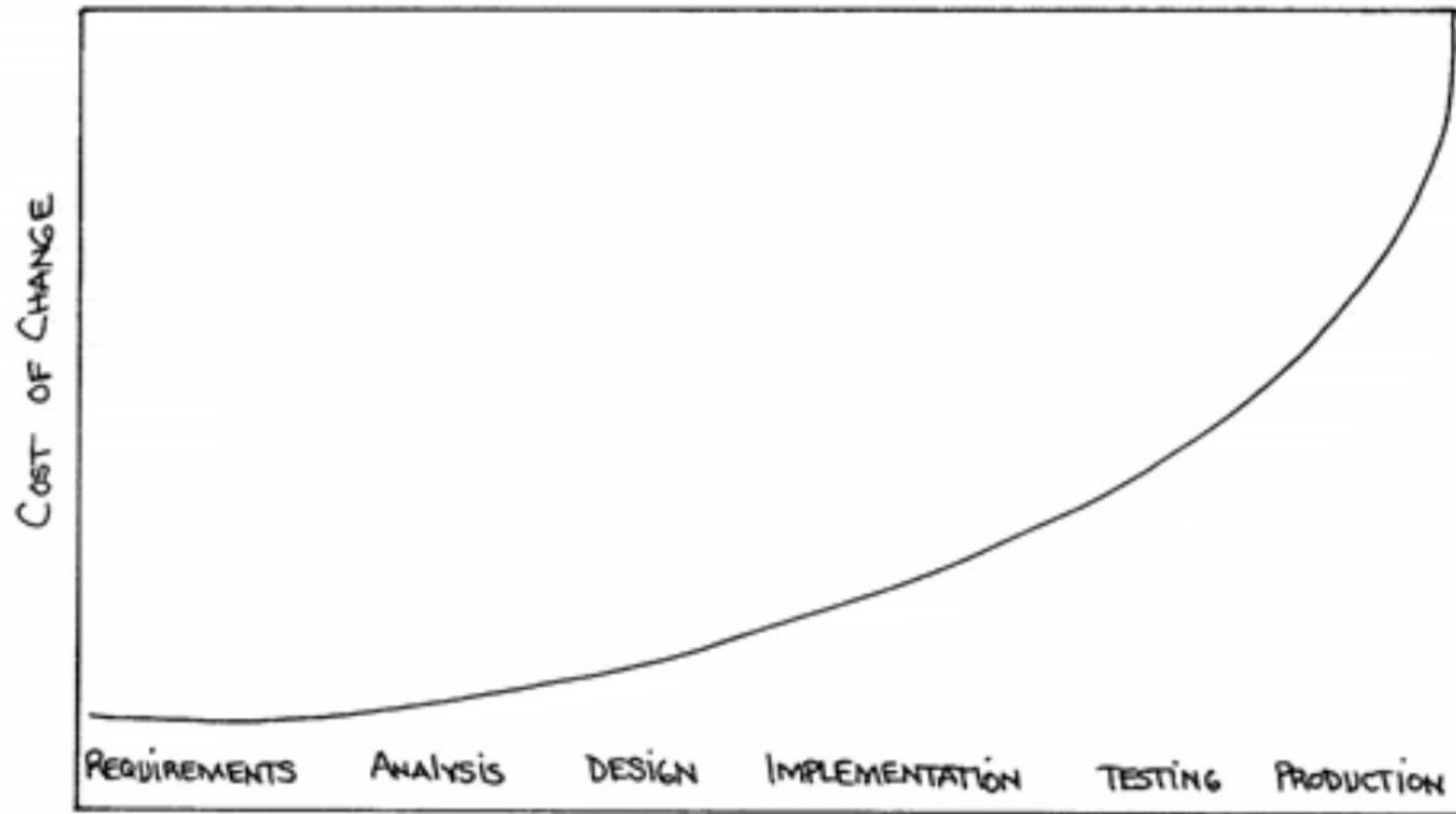
(2-3 semanas)



Barry
Boehn

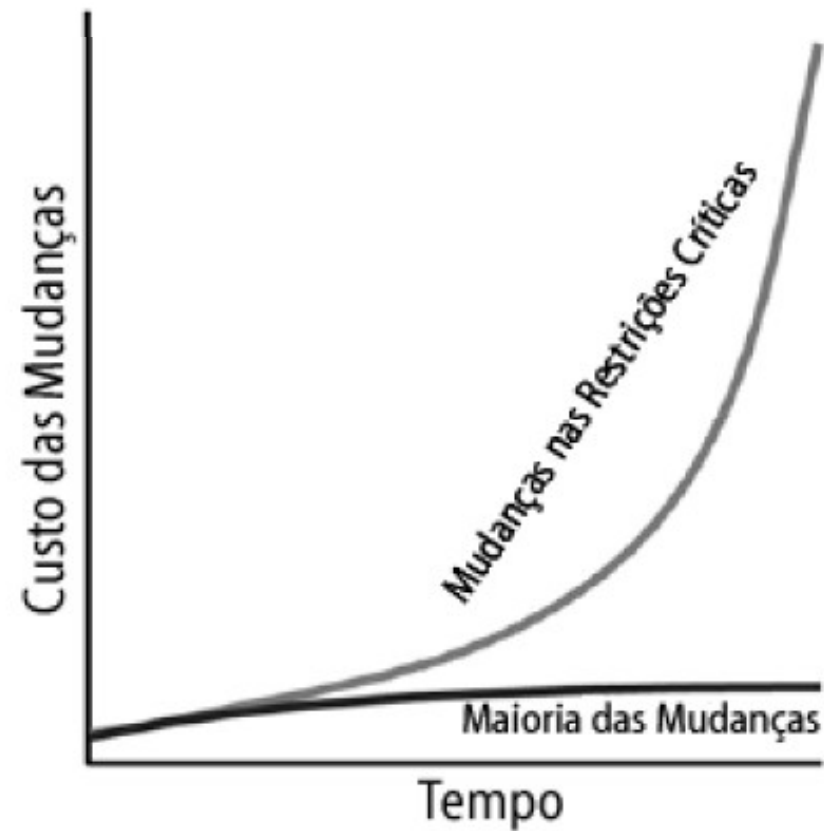


CUSTO DA MUDANÇA



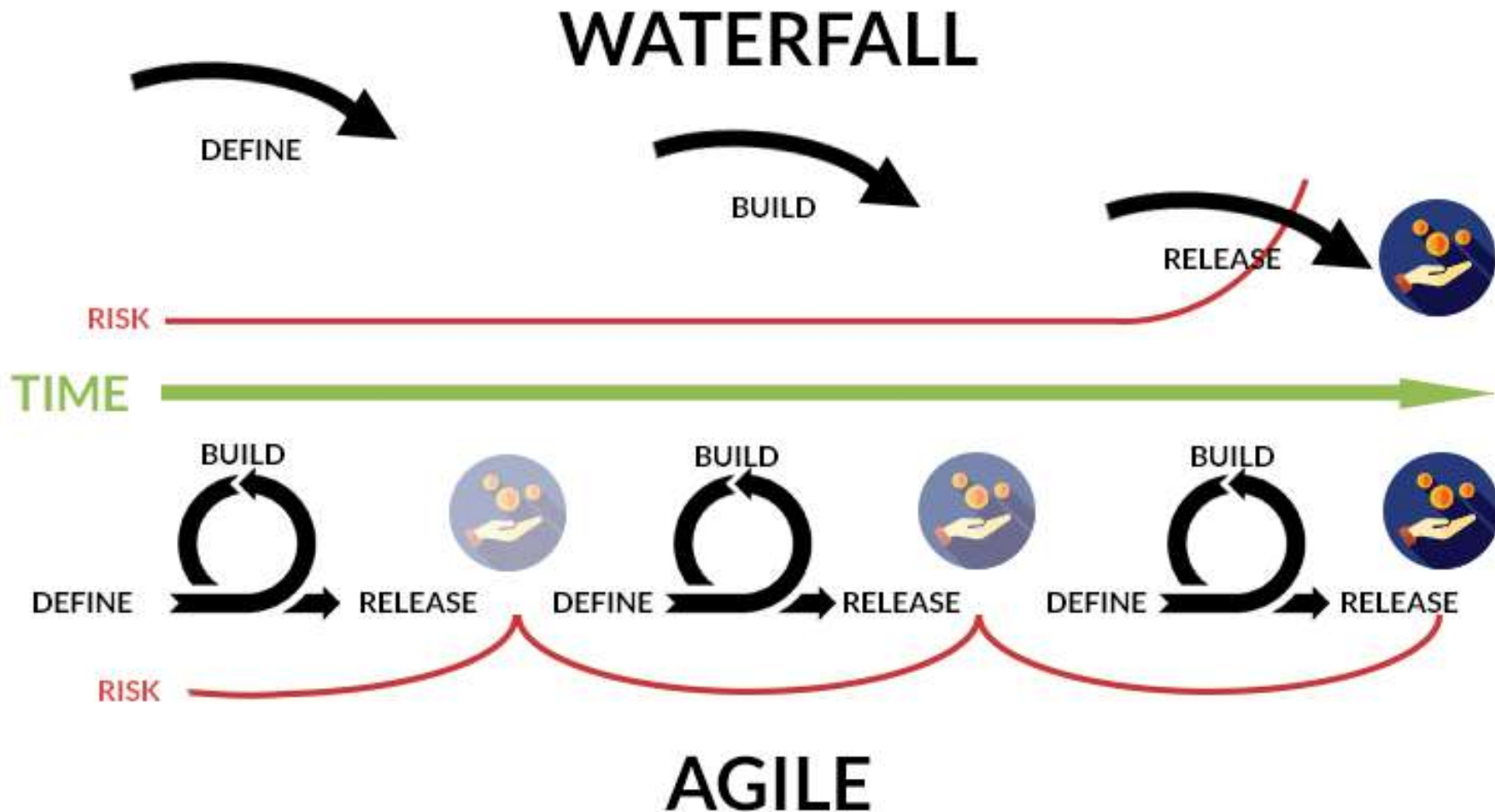
A tendência de software é o custo de mudança aumentar ao longo do tempo

METODOLOGIAS ÁGEIS - MUDANÇAS



PLANEJANDO

O planejamento é mais fácil com lançamentos frequentes



Aceite.

As estimativas são quase sempre erradas!



Como as estimativas são afetadas pelo tamanho da especificação

Spec



117 hrs

Same spec - more pages

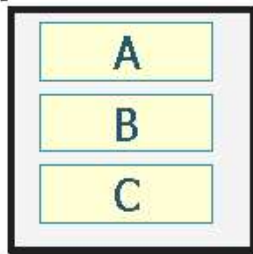


173 hrs

Fonte: Como evitar o impacto de informações irrelevantes e enganosas nas suas estimativas de custo, seminário de estimativa de laboratórios de pesquisa Simula, Oslo, Noruega, 2006

Como as estimativas são afetadas por informações irrelevantes

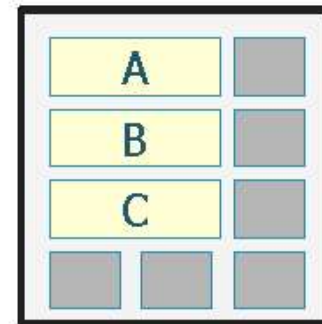
Spec 1



20 hrs



Same spec
+ irrelevant details



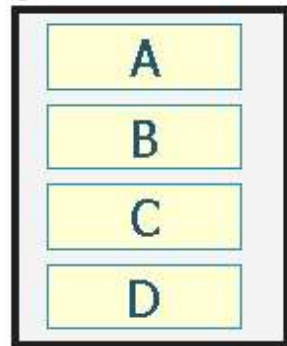
39 hrs



Fonte: Como evitar o impacto de informações irrelevantes e enganosas nas suas estimativas de custo, seminário de estimativa de laboratórios de pesquisa Simula, Oslo, Noruega, 2006

Como as estimativas são afetadas por requisitos extras

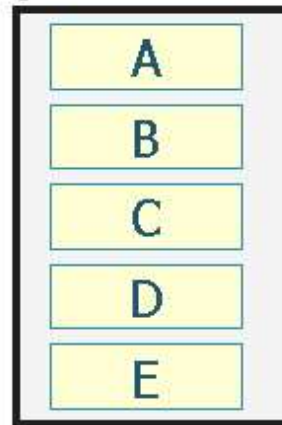
Spec 1



4 hrs



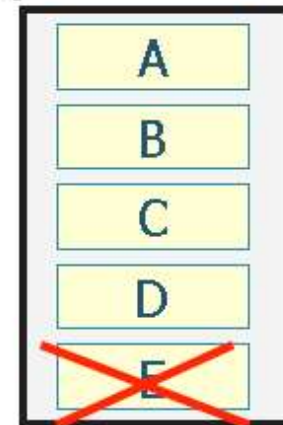
Spec 2



4 hrs



Spec 3



8 hrs



Fonte: Como evitar o impacto de informações irrelevantes e enganosas nas suas estimativas de custo, seminário de estimativa de laboratórios de pesquisa Simula, Oslo, Noruega, 2006

Como as estimativas são afetadas por ancoragem

Spec



456 hrs



Same spec



500 hrs
Never mind me

555 hrs



Same spec



50 hrs
Never mind me

99 hrs



Fonte: Como evitar o impacto de informações irrelevantes e enganosas nas suas estimativas de custo, seminário de estimativa de laboratórios de pesquisa Simula, Oslo, Noruega, 2006

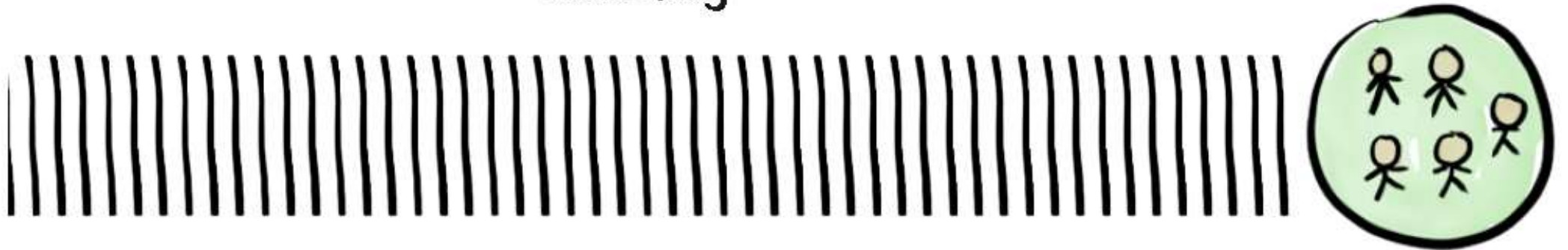
Velocidade

para conhecer o futuro, você precisa conhecer o passado



Planejamento de release com base na velocidade

Backlog



Planejamento de release com base na velocidade

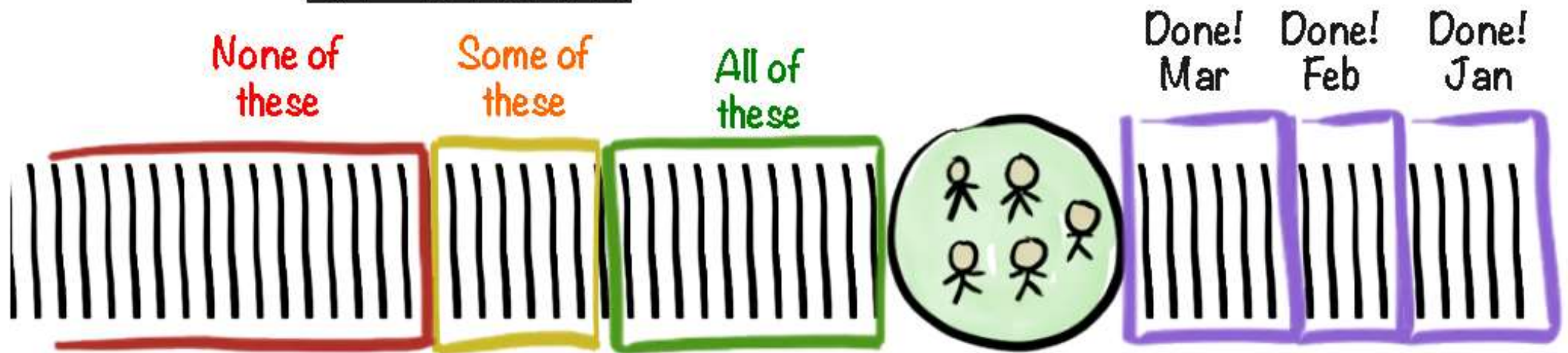


Planejamento de release com base na velocidade



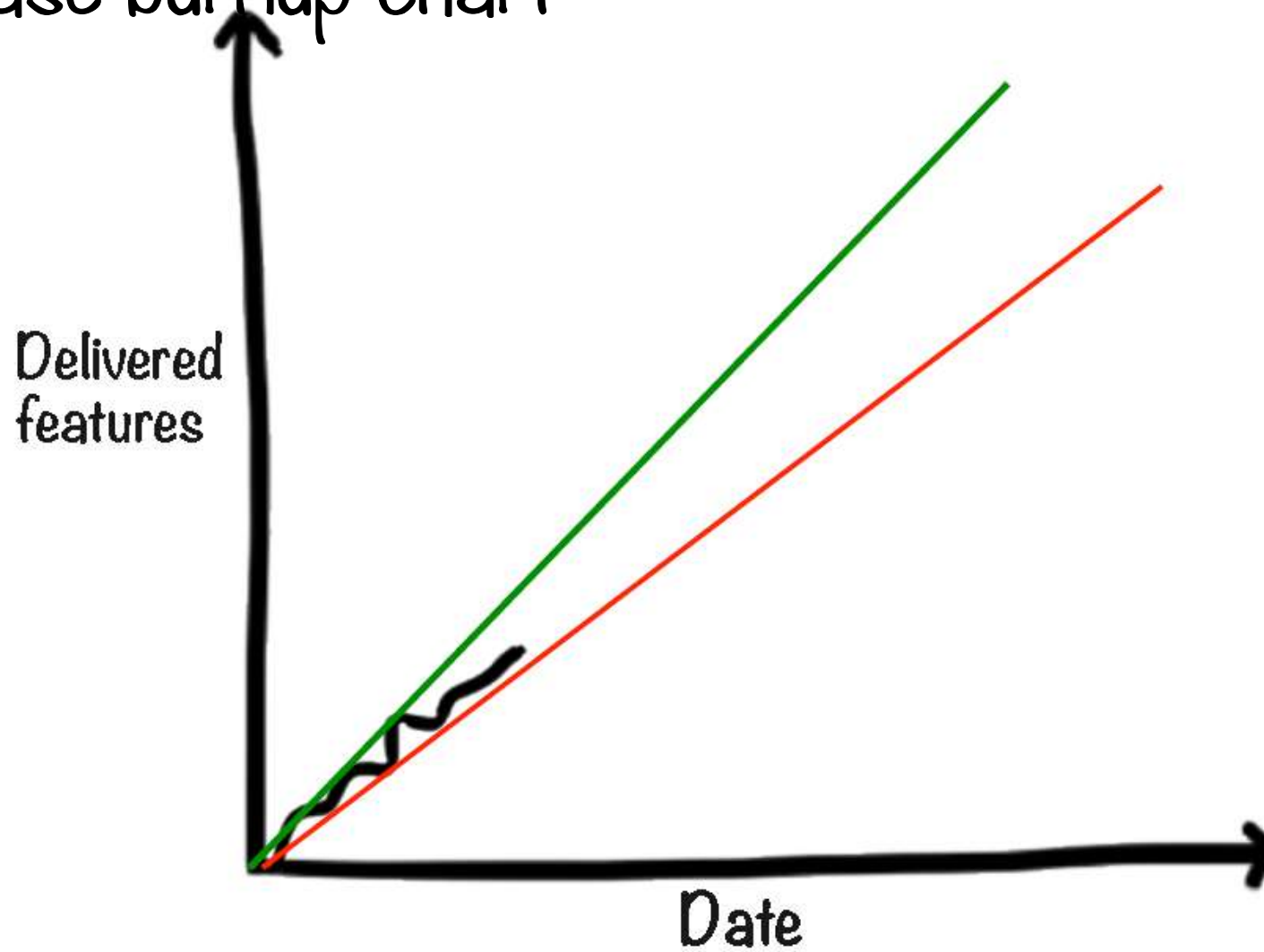
Planejamento de release com base na velocidade

Q2 forecast

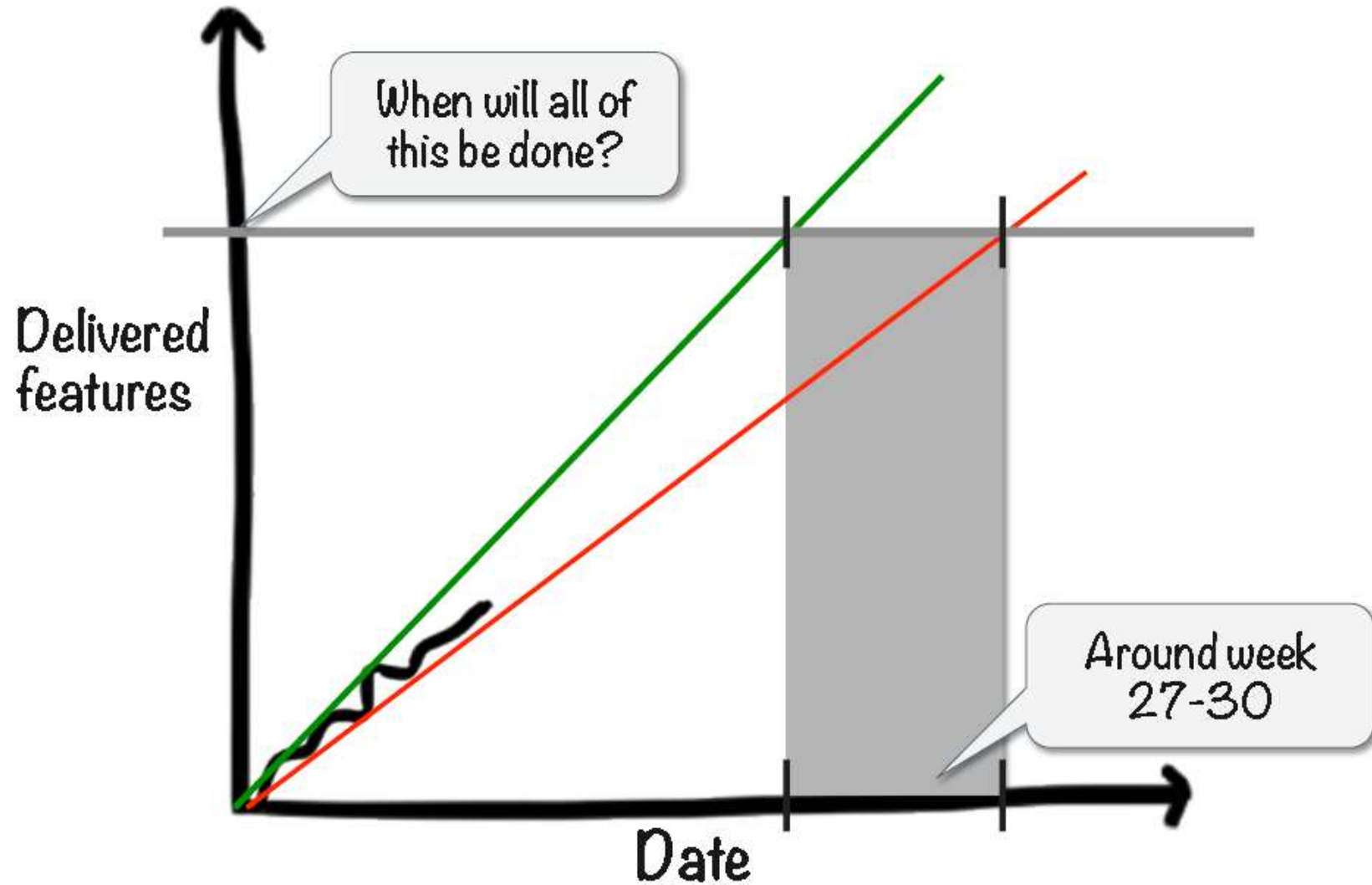


MÉTRICAS

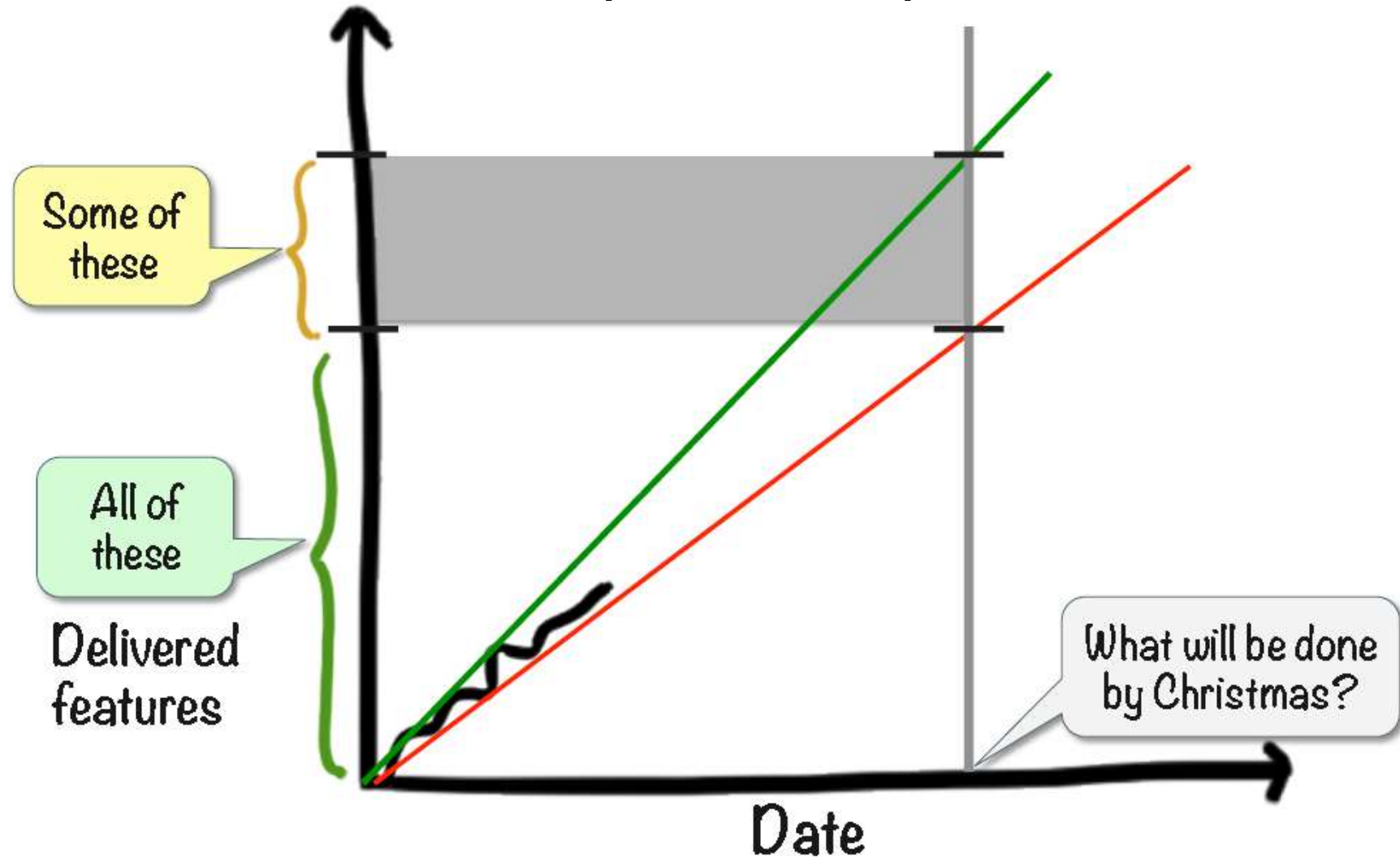
Release burnup chart



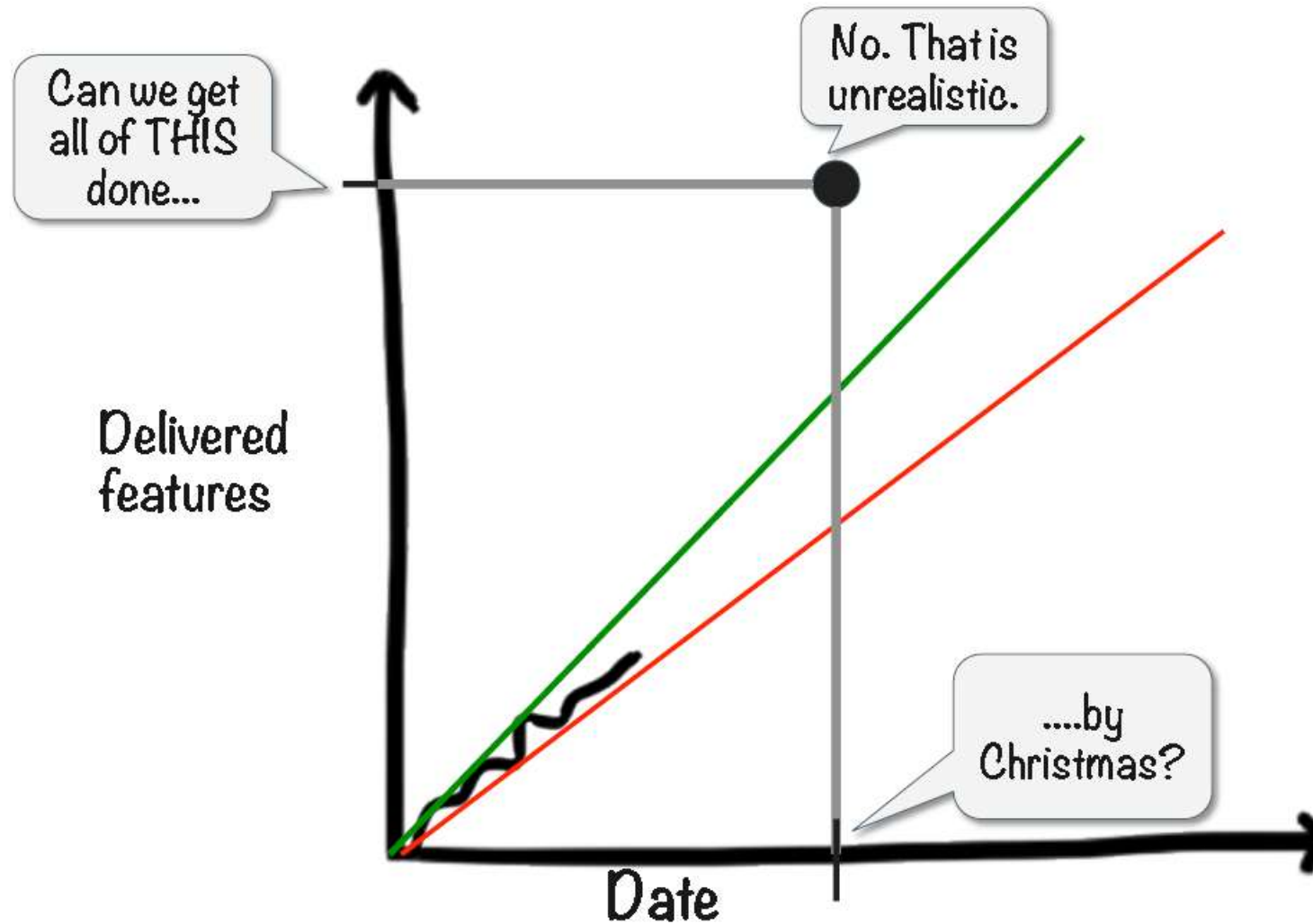
Estimativas com tempo fixo



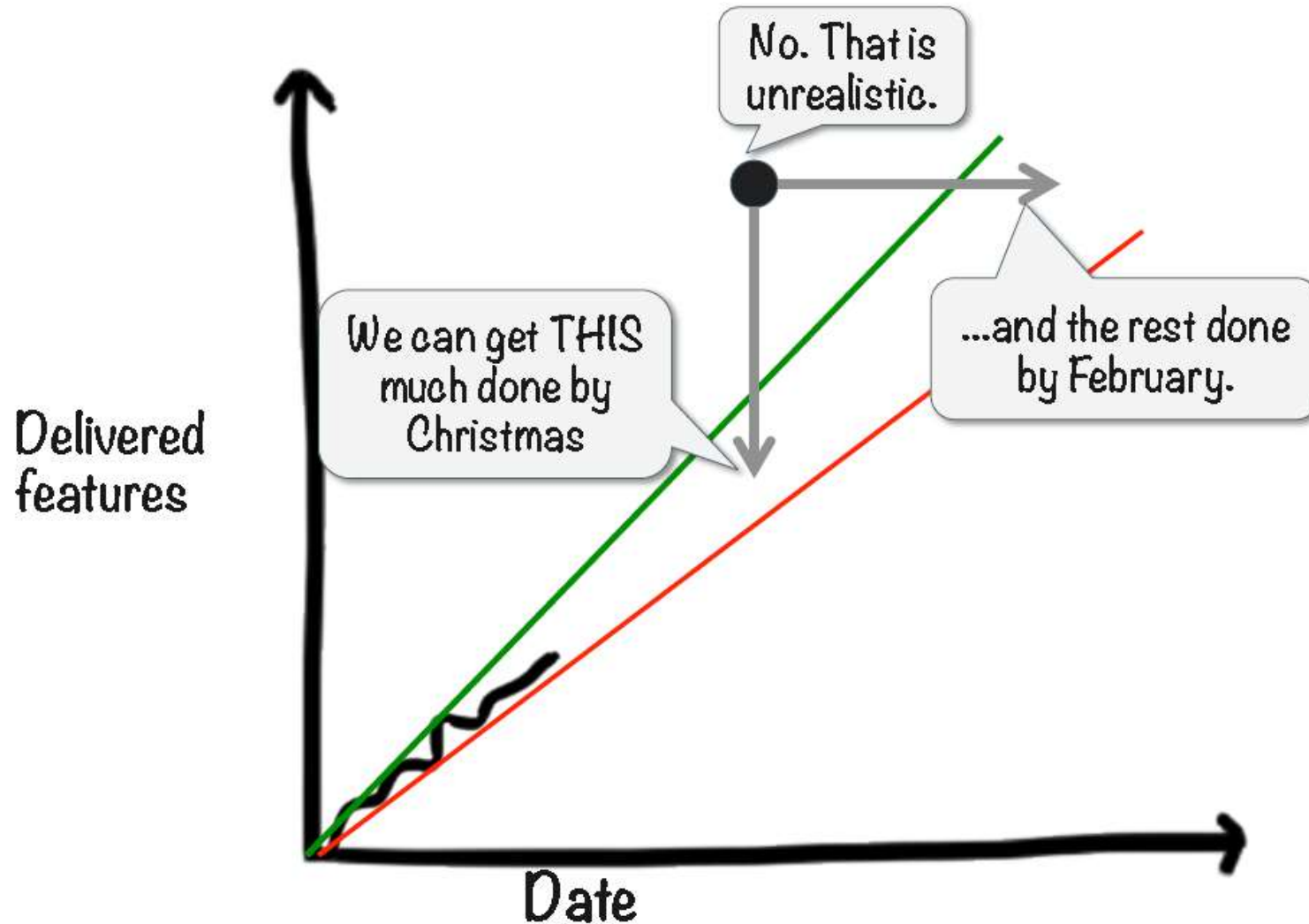
Estimativas com tempo e escopo fixos



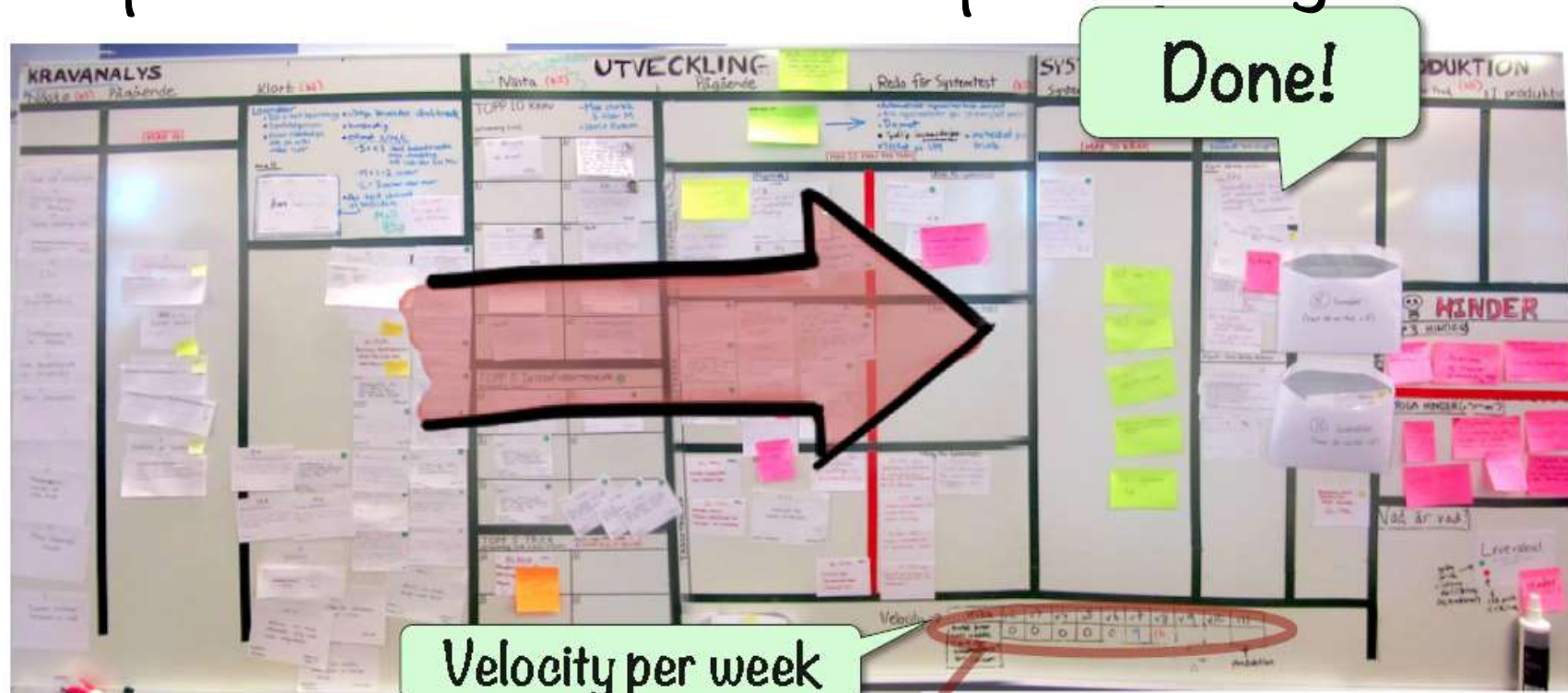
Estimativas com tempo e escopo fixos



Estimativas com tempo e escopo fixos



Exemplo: Medindo a velocidade pela contagem de cartões



Velocity per week

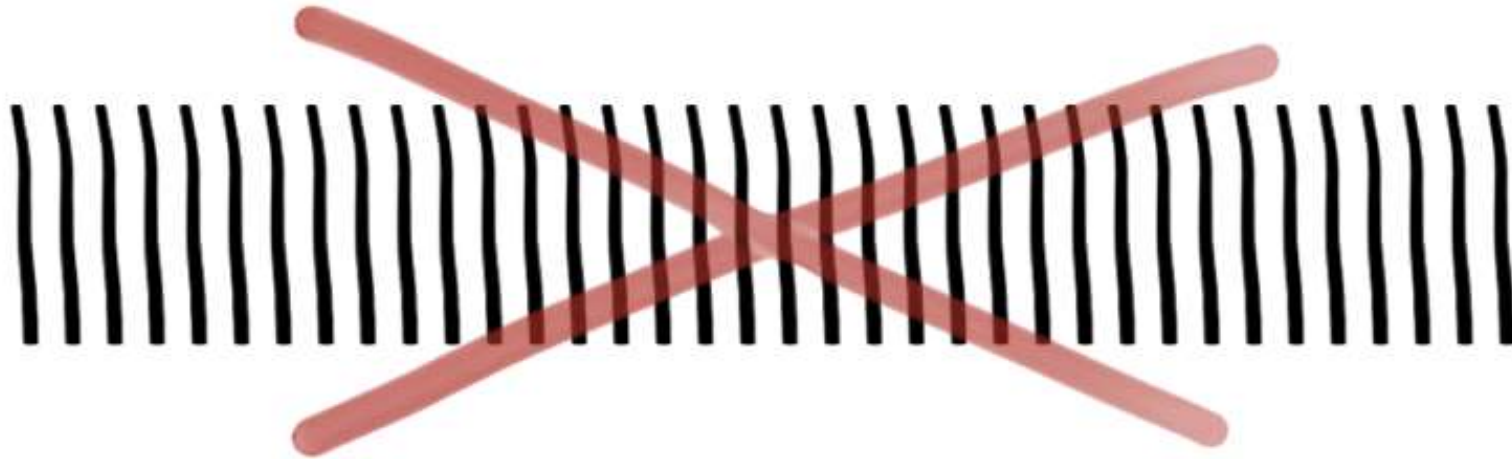
Vecka	v10	v11	v12	v13	v14	v15	v16	v17	v18
Antal nya funktioner som nått till 'Redo för Accept'	4	0	2	4	5	0			

↑ Prognos

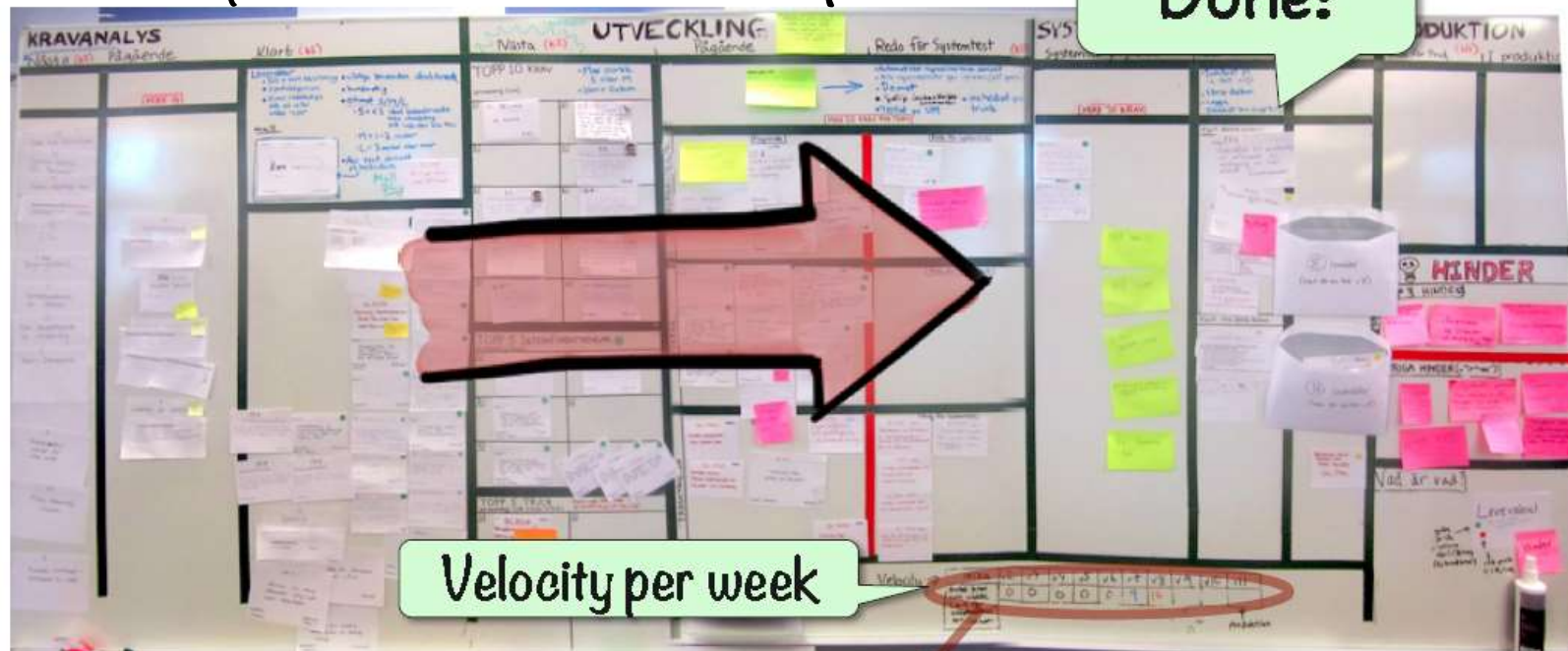
VEL

ESTIMANDO

Fato: as funcionalidades têm tamanhos diferentes



Opção 1: ignore a diferença de tamanho.
Isso se equilibra com o tempo.

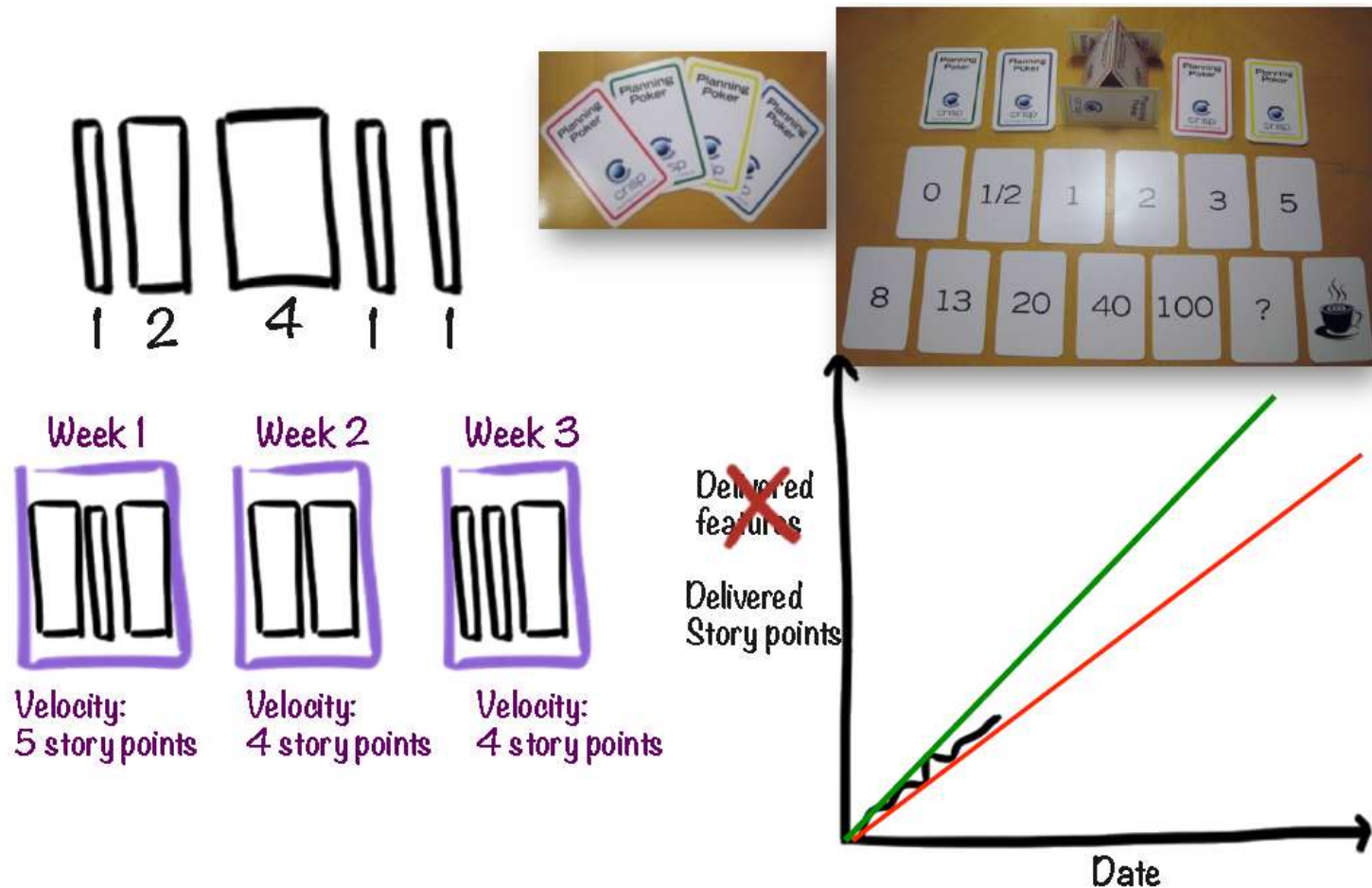


Vecka	v10	v11	v12	v13	v14	v15	v16	v17	v18
Antal nya funktioner som nått till 'Redo för AcTest'	4	0	2	4	5	0			

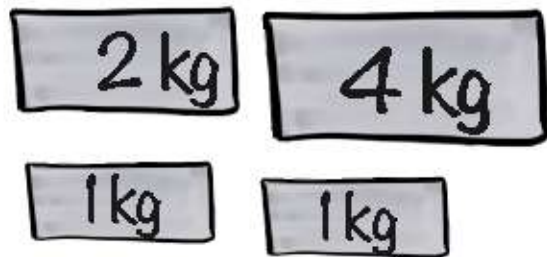
↑
Prodsättn.

VEL

Opção 2: Estime o tamanho relativo da funcionalidade.



Duas perguntas diferentes: tamanho e tempo



1: What is weight of each stone?

2: What is our delivery capacity?

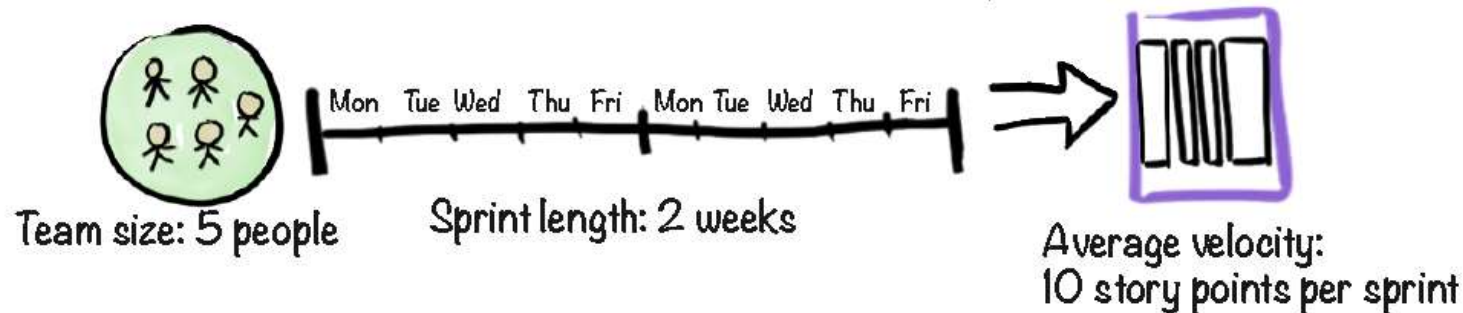
200 kg / hour



Estratégia de estimativa ágil

- Não estime tempo.
 - Estime o **tamanho relativo** das features.
 - Meça a velocidade por sprint.
 - **Derive** o plano de release.
- (Regra do Scrum) Estimativas feitas pelas pessoas **que farão o trabalho**.
 - Não pelas pessoas que querem o trabalho feito.
- Estime e reestime continuamente durante o projeto
 - Não confie nas estimativas iniciais
- Prefira comunicação verbal ao invés de especificações detalhadas e escritas.
- Evite falsa precisão
 - Melhor estar mais ou menos certo do que precisamente errado

Controle de custos sem relatórios de tempo



1 sprint = 200,000kr
(salary cost of 5 people for 2 weeks)

1 story point = 20,000kr
(200,000kr / 10 story points)

1 story point = 5 mandays
(50 mandays / 10 story points)

Better to be Roughly Right
than Precisely Wrong

Feature	Size	Cost	Cost
Delete user	3 sp	15 mandays	60,000kr
PDF export	2 sp	10 mandays	40,000kr
Outlook integration	8 sp	40 mandays	160,000kr

VALOR

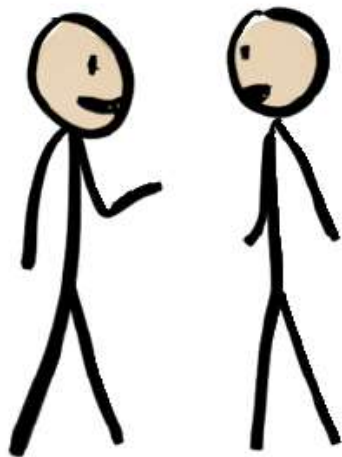
Features têm valor diferente (e o valor é independente do tamanho)



Weight: 1 gram
Value: 100 000 kr



Weight: 2000 grams
Value: 5 kr



2 minute standup discussion (pair/ trio):

- Give a real-life example of a feature that is **small** and **very valuable**
- Give a real-life example of a feature that is **large** and **not very valuable**.

Maximize o valor, não as entregas



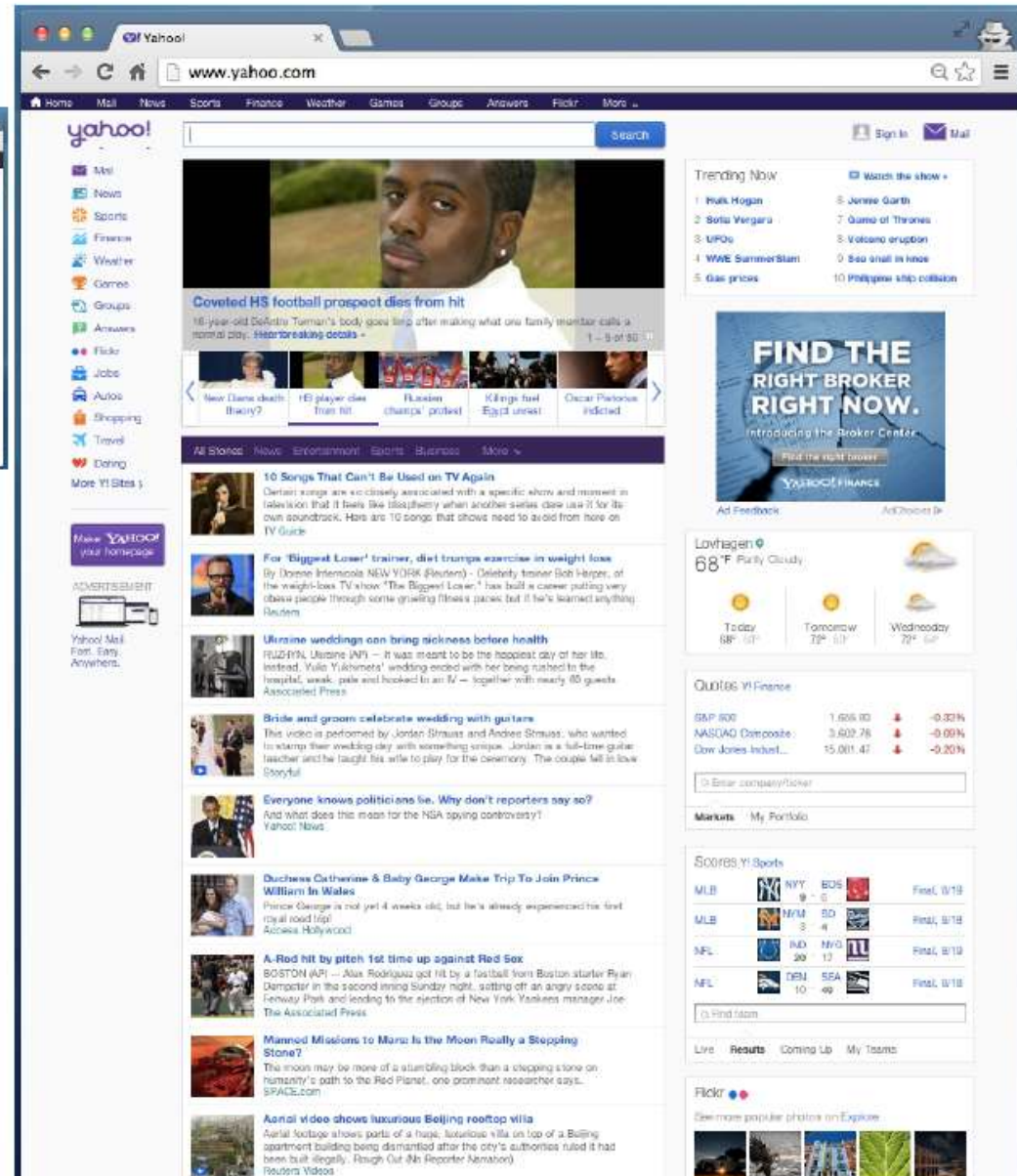
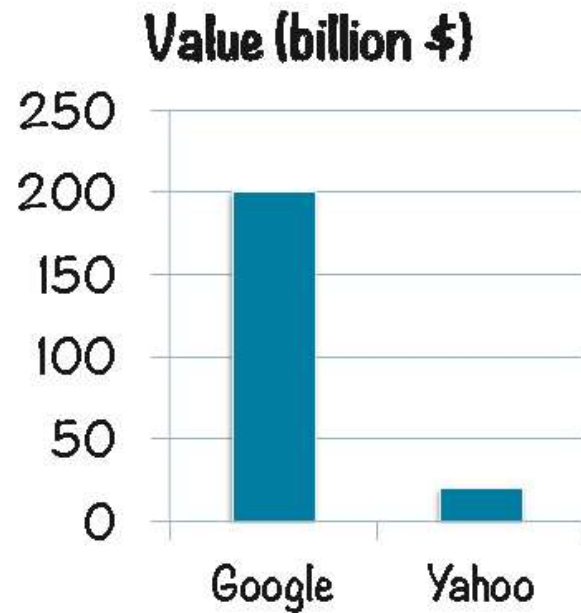
Menos é mais

A perfeição é alcançada, não quando não há mais nada a acrescentar, mas quando não há mais nada para tirar



Antoine de Saint-Exupéry

Google x Yahoo



REFERÊNCIAS

