

# **Metodologias Ágeis em Gerenciamento de Projetos**

Daniel de Amaral



# APRESENTAÇÃO

# Nossa agenda resumida

## Aula 1

- Apresentação da disciplina
- Conceituando GAP
- Valores e princípios da filosofia Ágil
- Aplicação do GAP
- Framework do GAP
- Discussão em grupo sobre GAP

## Aula 2

- Overview dos métodos ágeis
- Introdução ao Scrum
- O Framework Scrum

## Aula 3

- Planejando com Scrum
- Trabalho em grupo aplicando conceitos do Scrum

## Aula 4

- PMBoK e os métodos ágeis
- O papel do gestor na cultura do GAP
- Abordagem Ágil e os Contratos

+ Detalhes no Plano  
de Ensino da disciplina

# Visão do escopo de nossos encontros

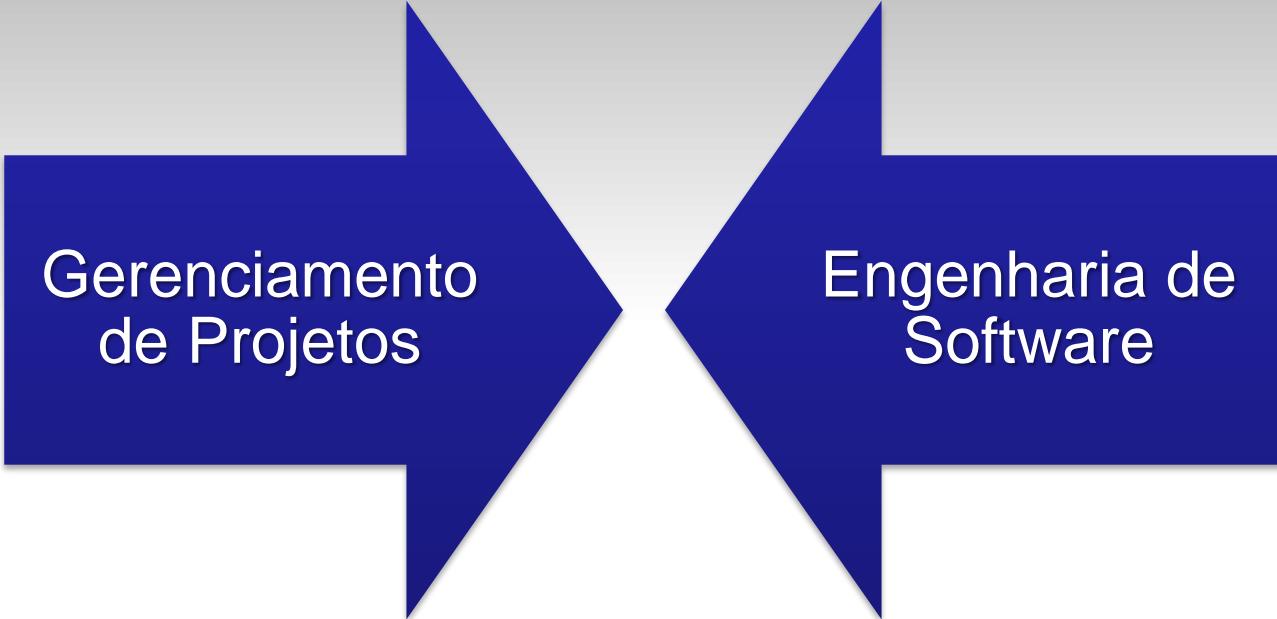
## Vamos discutir sobre

- Conceitos e principais paradigmas do Gerenciamento de Projetos na abordagem Ágil.
- Princípios e Valores que norteiam os Métodos Ágeis e o Gerenciamento Ágil de Projetos.
- Visão geral do Framework Scrum.
- Práticas de gestão da abordagem ágil.
- Papel do gestor em um ambiente de Agilidade.

## Não vamos discutir sobre

- Detalhes do Gerenciamento de Projetos “Tradicional” (foco das disciplinas PEP e GRTI)
- Práticas e técnicas utilizadas como ferramental de apoio as metodologias ágeis (TDD, Integração Contínua, etc.)
- Detalhes de Engenharia de Software relacionados ao conteúdo de nossa disciplina.
- Softwares utilizados como ferramental de apoio as práticas ágeis.

# AULA 01



Gerenciamento  
de Projetos

Engenharia de  
Software

# **GERENCIAMENTO DE PROJETOS**

# Relembrando

## O que é um Projeto?

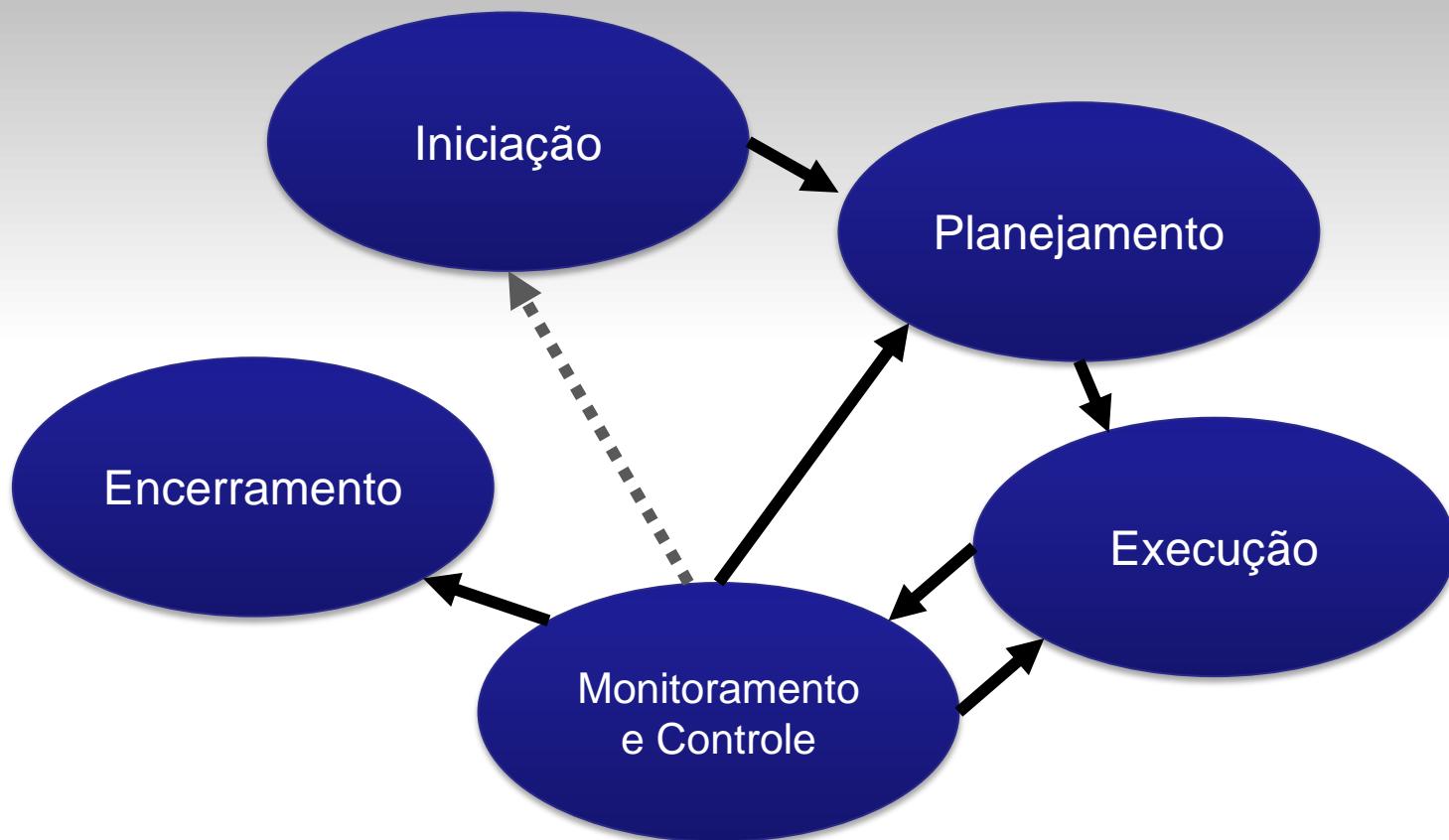
- Esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo. A sua natureza temporária indica um início e um término definidos (PMBOK, 4th edition)

# Relembrando

## O que é Gerenciamento de Projetos?

- Aplicação de conhecimentos, habilidades, ferramentas e técnicas às atividades do projeto a fim de atender aos seus requisitos (PMBOK, 4th edition)

# Relembrando Grupo de Processos PMBOK



# Relembrando

## Áreas de conhecimentos GP (PMBOK)

Gerenciamento da  
Integração

Gerenciamento do  
Escopo

Gerenciamento do  
Tempo

Gerenciamento  
dos Custos

Gerenciamento do  
Qualidade

Gerenciamento  
dos Recursos  
Humanos

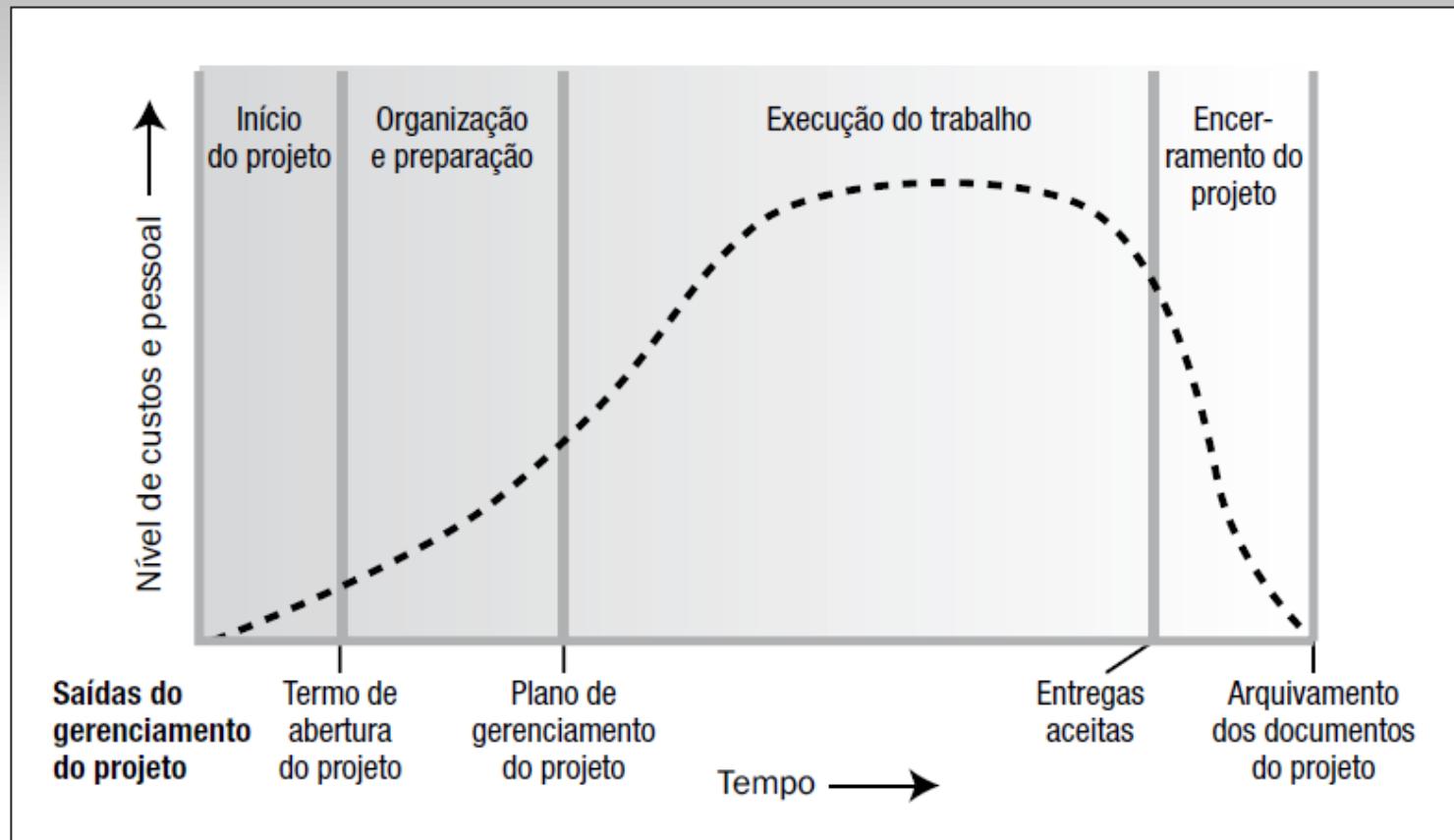
Gerenciamento  
das  
Comunicações

Gerenciamento  
dos Riscos

Gerenciamento  
das Aquisições

# Relembrando

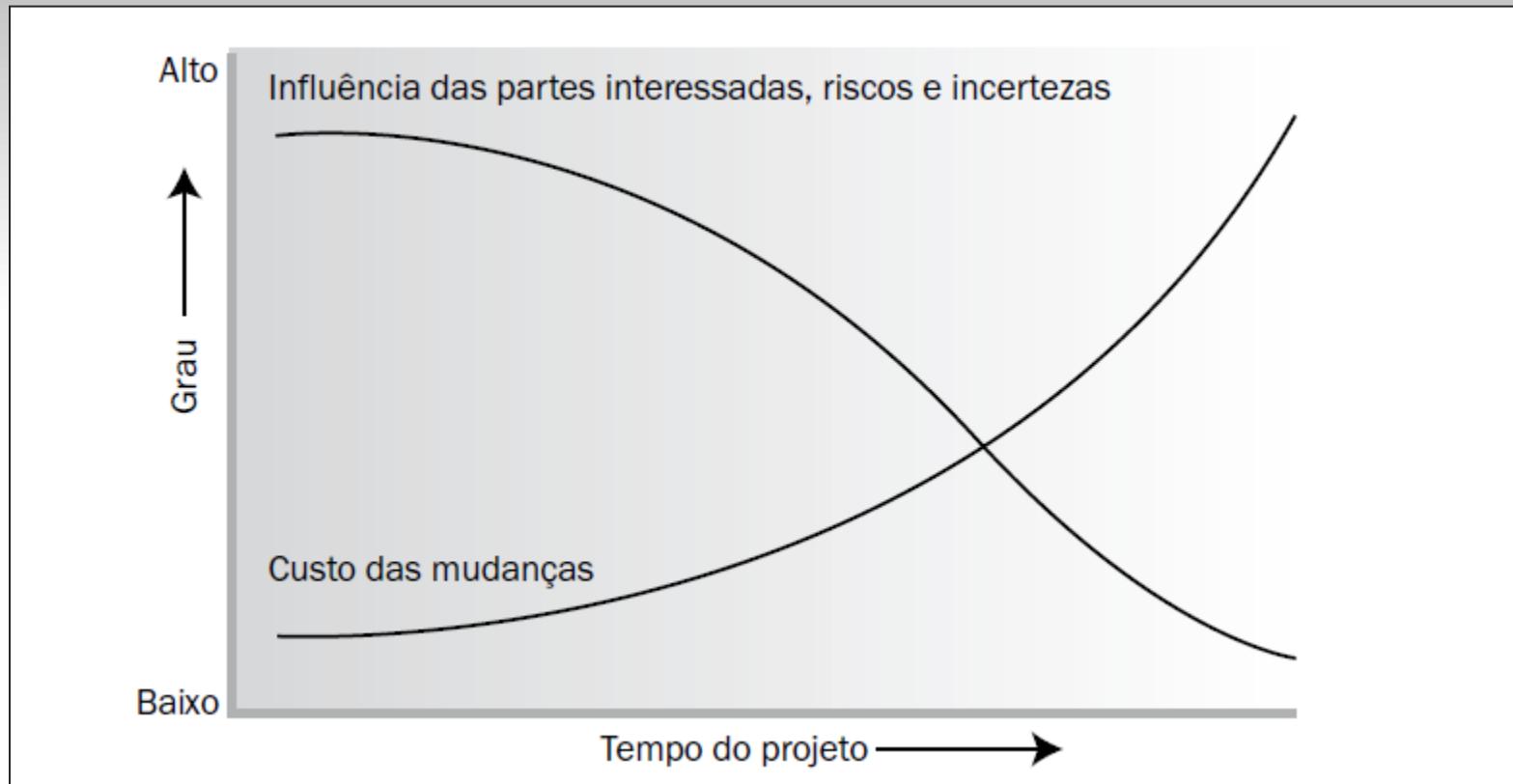
Nível típico de custos e pessoal  
de um projeto ao longo do seu ciclo de vida



Fonte: PMBOK, 2008

# Relembrando

## Impacto da variável com base no tempo decorrido do projeto



Fonte: PMBOK, 2008

# **PROJETOS DE SOFTWARE**

# Projetos e ciclos de vida de desenvolvimento de SW

- Nos projetos de tecnologia e desenvolvimento de software têm-se utilizado tradicionalmente a aplicação conjunta das práticas e processos do gerenciamento de projetos (PMBOK) com ciclos de vida lineares ou “Clássicos” de desenvolvimento de SW (ex.: Cascata / Waterfall).

# Modelos “Clássicos” de desenvolvimento de Software

- Cascata (*Waterfall*)

Definição dos requisitos de Sistema

Definição dos requisitos de Software

Análise

Design

Codificação

Testes

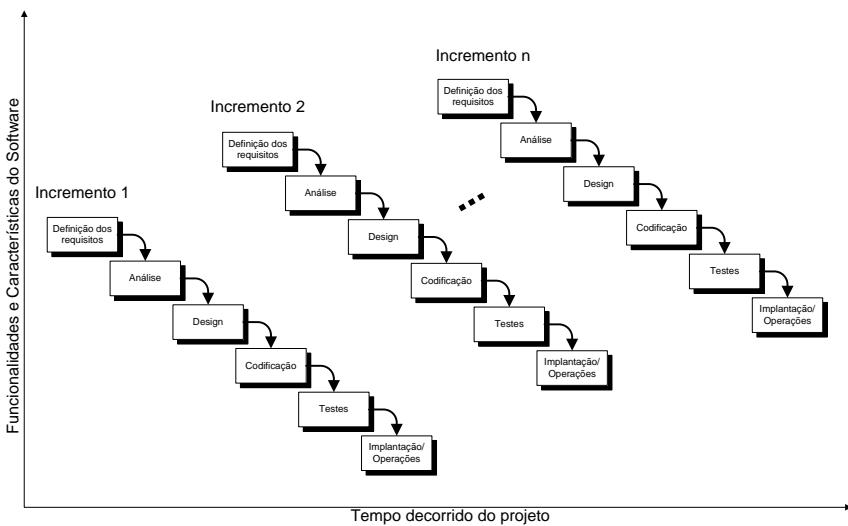
Implantação/  
Operações

Parênteses em defesa ao Dr. Royce : em seu artigo original ele já previa um ciclo iterativo entre as fases do modelo!

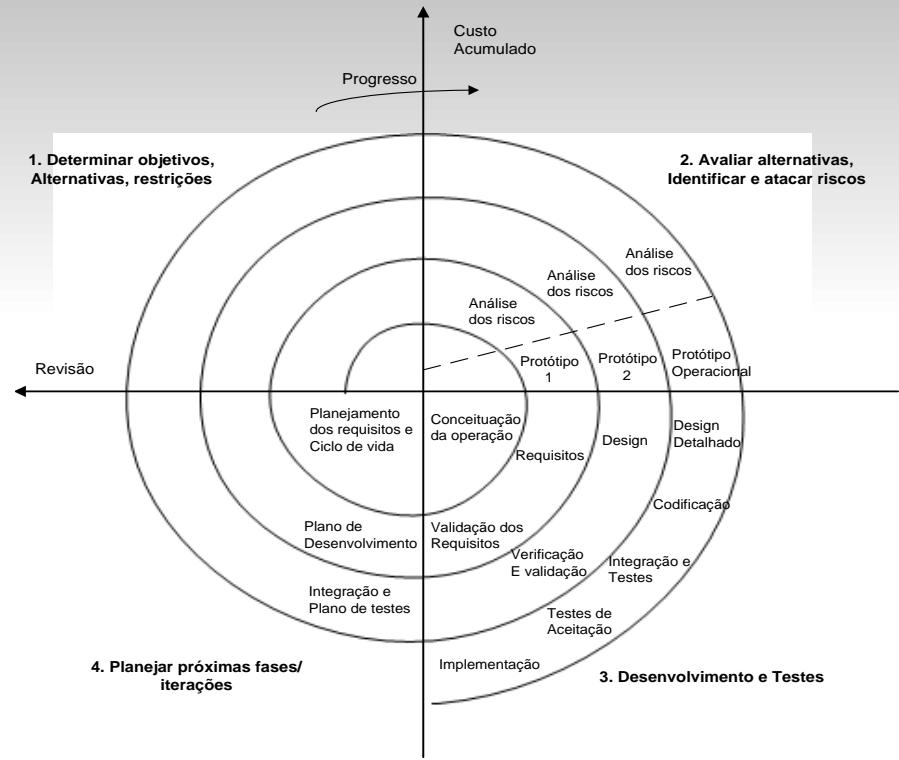
Fonte: ROYCE, 1970

# Modelos “Clássicos” de desenvolvimento de Software

- Incremental
- Espiral



Fonte: PRESSMAN, 2006



Fonte: BOEHM, 1988

# Modelos “Clássicos” de desenvolvimento de Software

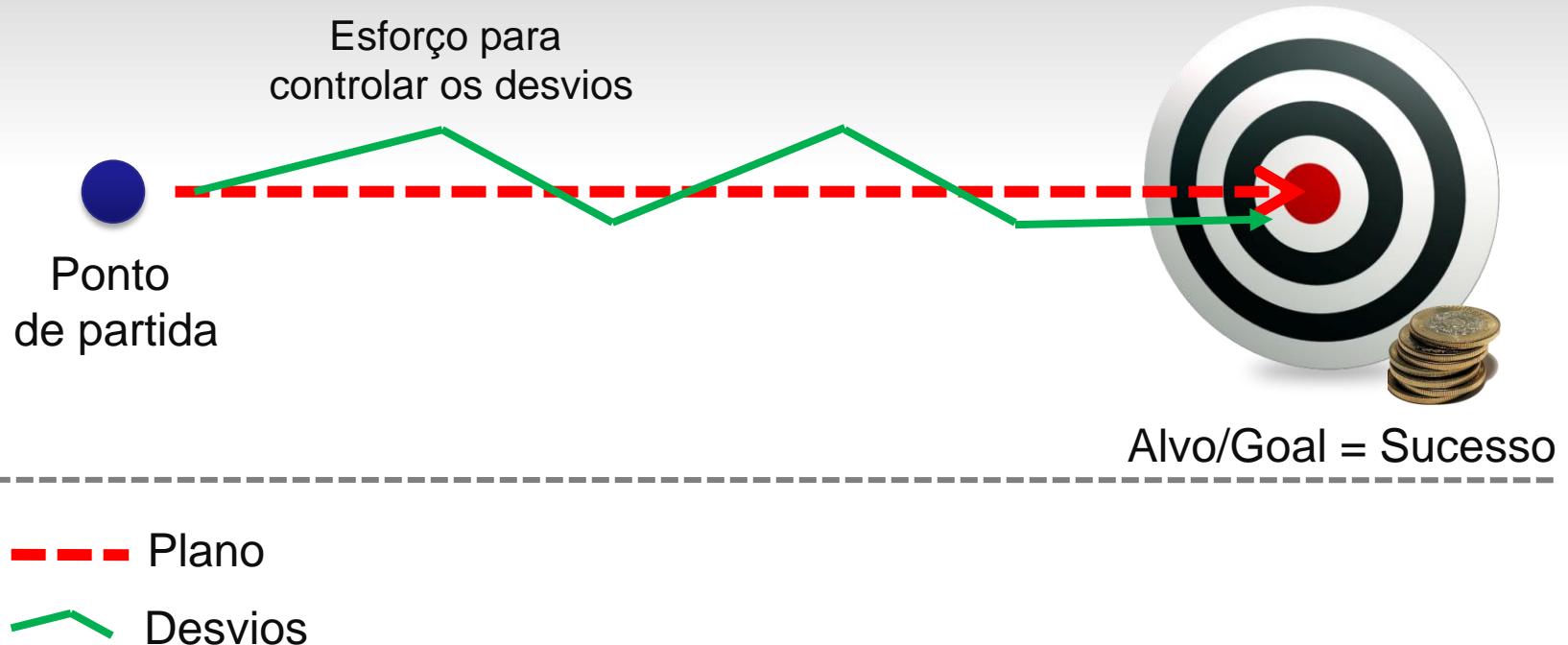
- Cria-se o plano de forma detalhada e todo o projeto é guiado através deste plano (“*Plan-Driven Process*”).
- O projeto de software é desenhado em todos seus detalhados antes de sua implementação (“*Big Design Up Front*”).
- Toda mudança é controlada de maneira formal (CCB).

# Modelos “Clássicos” de desenvolvimento de Software

- Através do rigor de planejamento busca-se previsibilidade e controle sobre as entregas a serem realizadas ao fim do projeto.
- Busca-se a execução através de um processo prescritivo.

# Determinando o objetivo

- Fixa-se o plano, e todo esforço busca controlar os possíveis desvios...



# Porém...

- O alvo é fixo?
  - A indústria de Software opera sob constante mudança (ambientes com alto grau de inovação).
  - Os processos do chamado Gerenciamento de Projetos “Tradicional” freqüentemente demonstram-se muito burocráticos e não atendem a natureza dinâmica do desenvolvimento de softwares.
  - Neste contexto, surge a necessidade de abordagens que possam facilmente responder as mudanças.

# Porém...

- O alvo é fixo?
  - Natureza dos projetos de Software:
    - Software é intangível.
    - Descoberta.
    - Investigação.
    - Evolução dos requisitos ao longo do desenvolvimento do produto.
    - Mudança!
  - Projetos complexos acabam demandando uma abordagem empírica.

# O alvo é fixo?



# **UMA REALIDADE A SE PENSAR**

# Project Success is Rare



Source: *Extreme Chaos, The Standish Group International, Inc., 2004, 2006, 2009*



Average cost overrun:

45%

Time overrun:

63%

Functionality delivered on average:

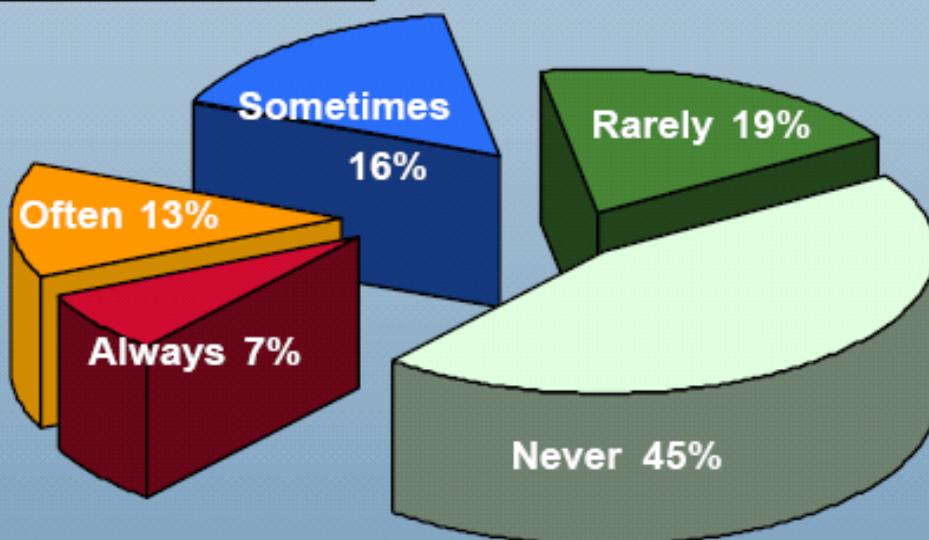
67%

Standish Group

## Features / Functions Used in a Typical System

**Often / Always  
Used: 20%**

**Rarely / Never  
Used: 64%**



*Standish Group Study Reported at XP2002 by Jim Johnson, Chairman*

# O que define sucesso em seus projetos?

- É apenas entregar no prazo e dentro do orçamento previsto?
- E quanto a satisfação do cliente?

# ÁGIL

# Agilidade

- a.gi.li.da.de  
*sf (lat agilitate)*
1. Qualidade do que é ágil.
  2. Desembaraço, leveza, presteza de movimentos.
  3. Mobilidade, perspicácia, vivacidade.

Fonte: [Michaelis Online](#)

# Em nosso contexto, o que é Agilidade?

- Habilidade de criar e responder a mudanças de forma a manter a lucratividade em um turbulento ambiente de negócios (HIGHSMITH, 2002)

# Desenvolvimento Ágil



# Breve visão histórica dos últimos 40 anos

**Anos 70**

- Artigo “Managing the Development of Large Software Systems” publicado por Dr. Winston W. Royce.
- Evolutionary processes por Tom Gilb.

**Anos 80**

- “Spiral Model of Software Development and Enhancemen” de Barry Boehm.
- Publicado o livro “The Mythical Man Month” de Fred Brooks.
- PeopleWare por DeMarco and Lister.
- Artigo NNPDG article de Takeuchi e Nonaka.
- RUP Objectory v1.0
- Publicado o Capability Maturity Model (CMM) e livro Managing the Software Process de Watts Humphrey's.

**Anos 90**

- Dynamic Systems Development Methodology (DSDM).
- Crystal Methodologies de Alistair Cockburn.
- Primeiros passos do Scrum para projetos de SW por Jeff S. e Ken Schwaber.
- Origens do Extreme Programming (XP) por Kent Beck.
- Feature Driven Development por Peter Coad e Jeff De Luca.

**2000's**

- XP ganhando “momentum”.
- Adaptive Software Development por Jim Highsmith.
- Publicação do Manifesto Ágil.
- Metodologias e práticas Ágeis crescendo de maneira significativa no mercado.

# Breve visão histórica dos últimos 40 anos

- Mesmo com todos os esforços em tornar o processo de desenvolvimento de software mais efetivo (RUP, etc.), a prática apresentava (ainda apresenta em muitos casos)...

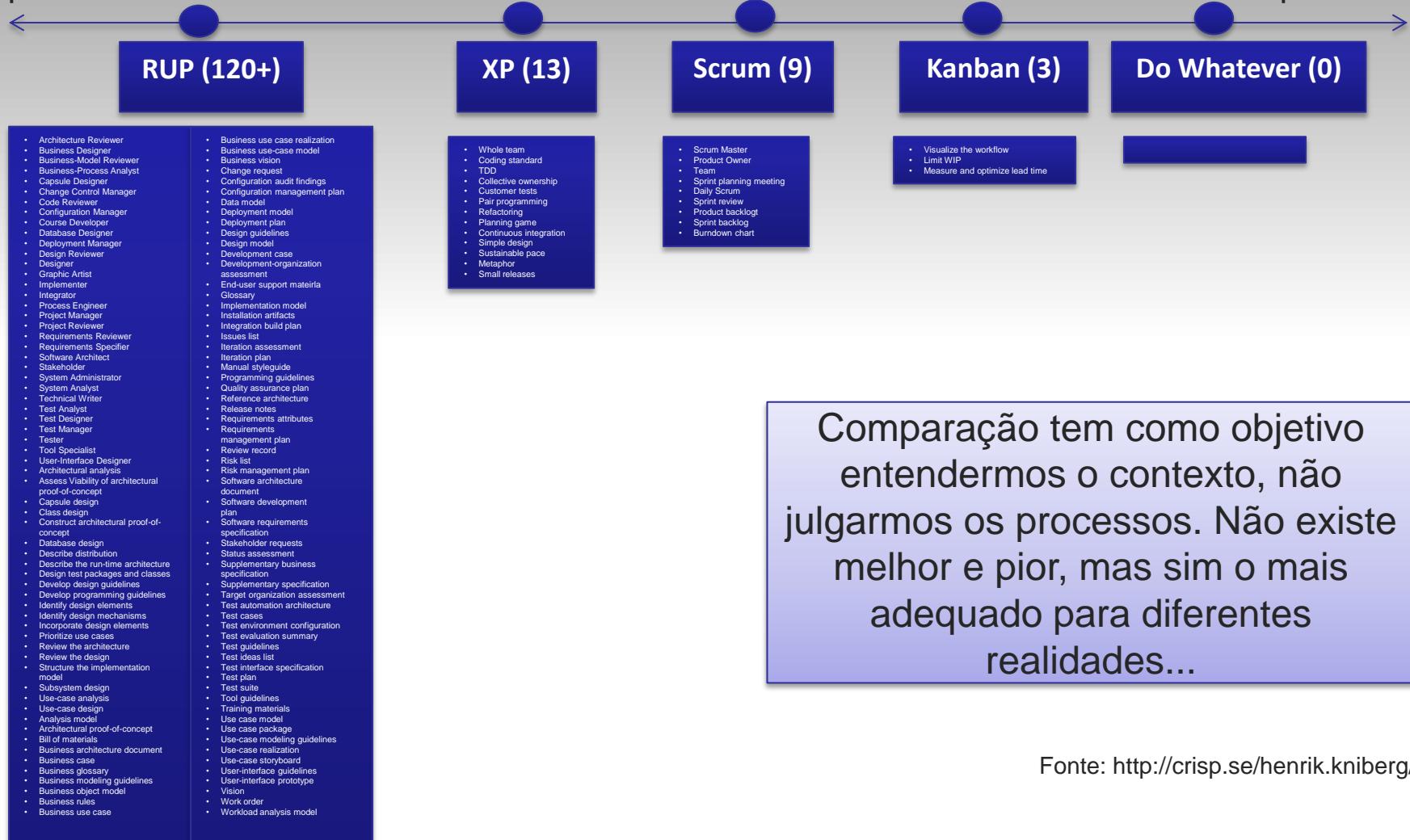


- Excesso de documentação
- Lentidão de resposta as mudanças
- Distanciamento entre o que o cliente quer (ou precisa) em relação ao que é entregue

# Processos Prescritivos vs Adaptativos

Mais prescritivos

Mais adaptativos



Comparação tem como objetivo entendermos o contexto, não julgarmos os processos. Não existe melhor e pior, mas sim o mais adequado para diferentes realidades...

Fonte: <http://crisp.se/henrik.kniberg/>

# Manifesto Ágil

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

**Indivíduos e suas interações** acima de processos e ferramentas

**Software funcionando** acima de documentação abrangente

**Colaboração do cliente** acima de negociação de contratos

**Resposta às mudanças** acima da execução de um plano

Isto é, ainda que haja valor nos itens à direita, valorizamos mais os itens à esquerda.

# 12 Princípios do Manifesto (1/2)

Nossa maior prioridade é satisfazer os clientes através de rápidas e contínuas entregas de software com valor agregado

Acolhemos mudança de requisitos, inclusive em etapas avançadas do desenvolvimento. Processos ágeis permitem mudanças que contribuem para a vantagem competitiva de seus clientes

Entregar software funcionando frequentemente, em algumas semanas ou meses, de preferência no menor tempo possível

Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto

Construa projetos através de indivíduos motivados. Dê à equipe um ambiente que atenda suas necessidades e confie em sua capacidade para realizar o trabalho

O método mais eficiente e efetivo de distribuir informação para e entre uma equipe de desenvolvimento é a comunicação face a face

Fonte: <http://agilemanifesto.org/>

# 12 Princípios do Manifesto (2/2)

Software funcional é a medida primária de progresso

Processos ágeis estimulam o desenvolvimento sustentável. Os patrocinadores, desenvolvedores, e usuários devem estar aptos a manter um ritmo constante indefinidamente

Atenção contínua na excelência técnica e num bom projeto/design aprimora a agilidade

Simplicidade - a arte de maximizar a quantidade de trabalho não feito - é essencial

As melhores arquiteturas, requisitos e projetos (designs) emergem de equipes auto-organizadas

Em intervalos regulares, a equipe reflete sobre como podem ser mais eficazes, então sintonizam e ajustam seus comportamentos desta maneira

Fonte: <http://agilemanifesto.org/>

# Metodologias Ágeis de Desenvolvimento de Software

## Extreme Scrum Programming (XP)

Crystal Methods

Adaptive Software Development (ASD)

Lean Development (LD)

Feature Driven Development (FDD)

Agile Modeling (AM)

Dynamic Systems Development Method (DSDM)

# Adoção dos Métodos Ágeis pelo mercado

- Estudo recente realizado pela Forrester Research Inc. (~1300 profissionais de TI entrevistados) aponta 35% dos entrevistados respondendo que seu processo de desenvolvimento está muito próximo/ligado aos métodos ágeis.

Fonte: [InfoWorld, 2010](#)  
[Forrester, 2010](#)

# Empresas utilizando métodos Ágeis (em pequena ou grande escala)

- Google
- Yahoo!
- Adobe
- HP
- IBM
- Microsoft
- Intel
- Cisco
- 3M
- Nasdaq
- Siemens
- Motorola
- Nokia
- Electronic Arts (EA Games)
- Departamento de Defesa Americano
- E muitas outras (grande penetração em pequenas e médias empresas)

# Gerenciamento Ágil de Projetos (GAP)

- Diante o advento das metodologias ágeis de desenvolvimento de software (mundo da engenharia de sw) o Gerenciamento de Projetos (mundo da gestão) é repensado para esta nova abordagem. Surge assim o chamado “Gerenciamento Ágil de Projetos”.

# Gerenciamento Ágil de Projetos (GAP)

- Conjunto de valores, princípios e práticas que auxiliam equipes de projeto a entregar produtos ou serviços de valor em um ambiente complexo, instável e desafiador Highsmith (2004)
- Nova plataforma de gerenciamento de projetos, aplicável a ambientes voláteis e desafiadores, sujeitos a mudanças freqüentes, em que o processo prescritivo e padronizado acaba não apresentando-se como a opção mais adequada Chin (2004)

# Declaração de Interdependencia (1/2)

Abordagens ágeis e adaptativas para conectar pessoas, projetos e valor.

Nós somos uma comunidade de líderes de projetos que são altamente bem-sucedidos em entregar resultados. Para alcançar esses resultados:

**Aumentamos o retorno sobre o investimento** fazendo do fluxo contínuo de valor o nosso foco.

**Entregamos resultados confiáveis** engajando clientes em interações frequentes e em propriedade compartilhada.

**Esperamos a incerteza e a administrarmos** através de iterações, antecipação e adaptação.

Fonte: <http://pmdoi.org/>

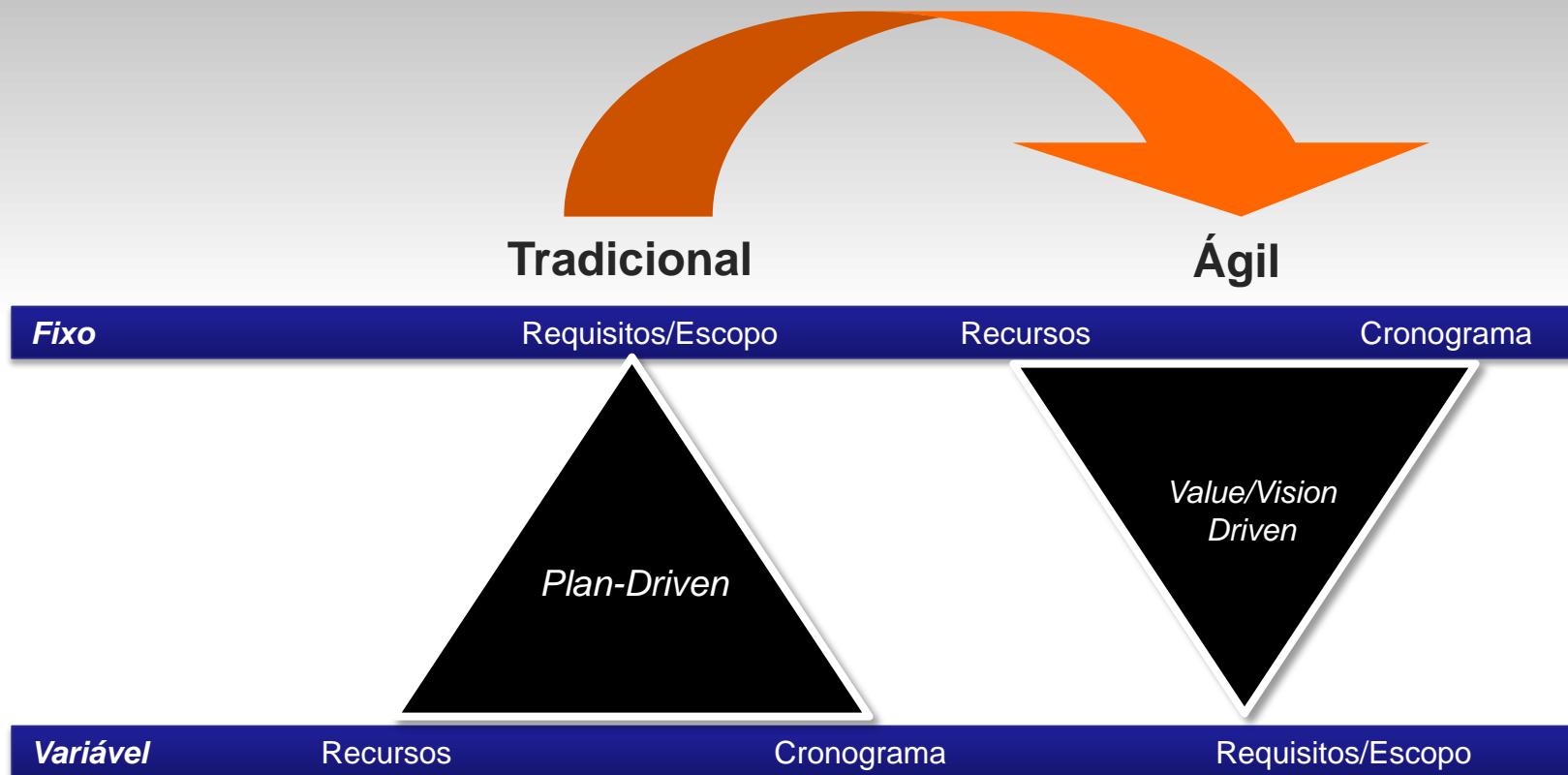
# Declaração de Interdependencia (2/2)

- **Libertamos a criatividade e a inovação** reconhecendo que os indivíduos são a fonte de valor mais importante, e criando um ambiente no qual eles podem fazer a diferença.
- **Promovemos o desempenho** através da responsabilidade do grupo pelos resultados e responsabilidade compartilhada pela efetividade da equipe.
- **Melhoramos a efetividade e confiabilidade** através de estratégias, processos e práticas específicos para cada situação.

Fonte: <http://pmdoi.org/>

# Uma mudança de paradigma

## Waterfall x Agile



Fonte: Sliger & Broderick, 2008

# Processo Definido e Empírico

## Diferença das abordagens

Eu conheço todos os detalhes sobre o que deve ser feito (detalha-se o plano e requisitos)

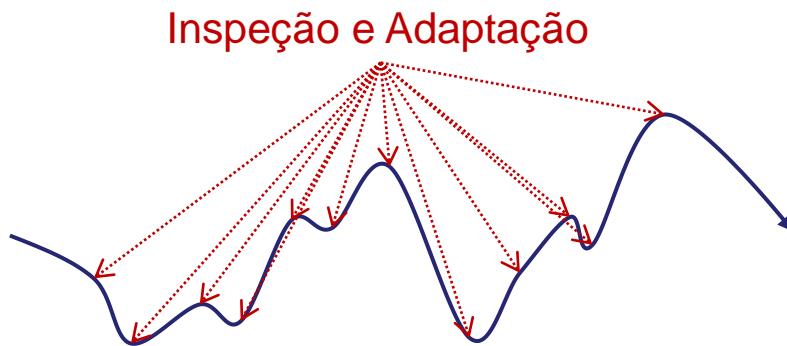
Termina com todos os requisitos entregues

Guiado por um plano  
(Plan-Driven)

Eu conheço a visão do produto/projeto (inicia com visão geral definida e requisitos de maior valor)

Termina ao Definir que os *goals* da visão foram alcançados

Empírico  
Guiado por uma visão de valor

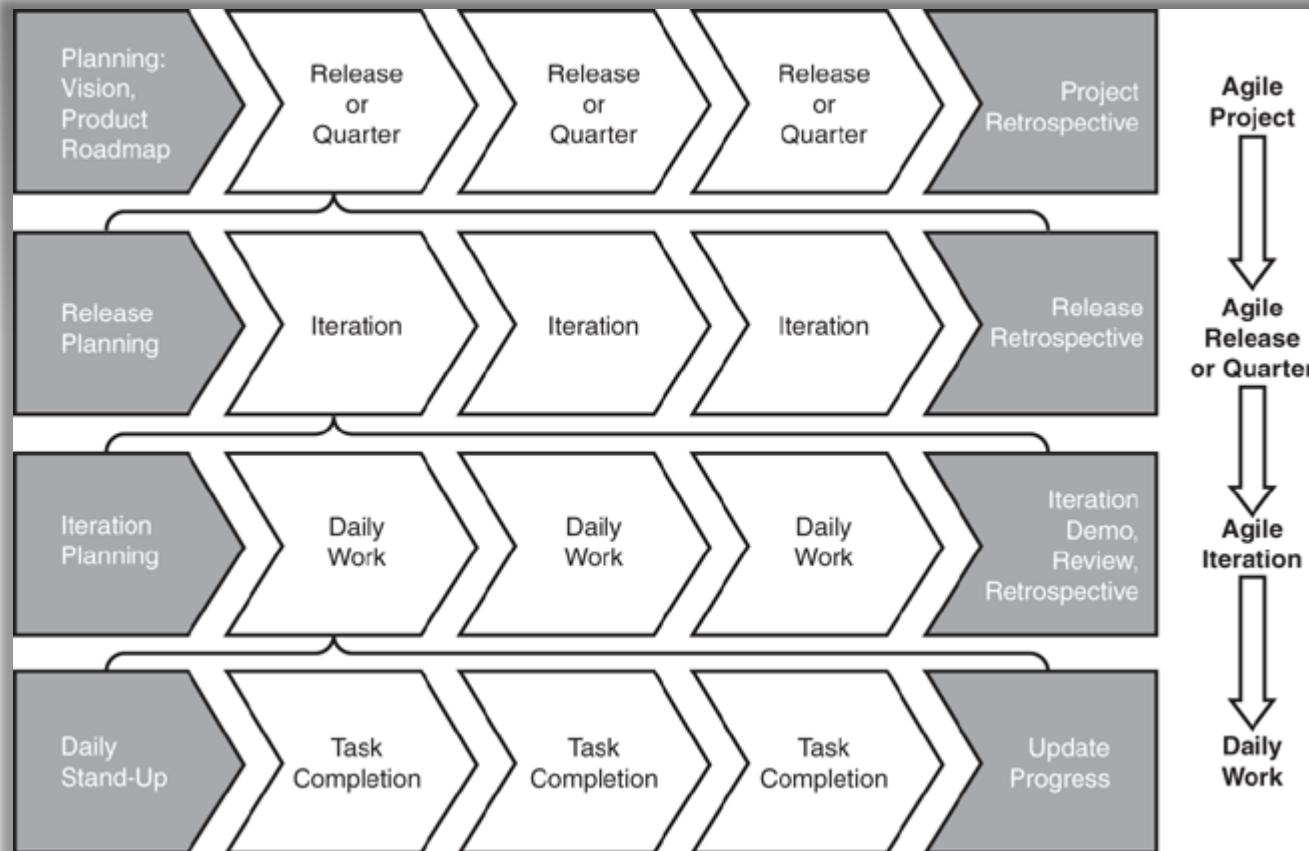


# Framework do GAP



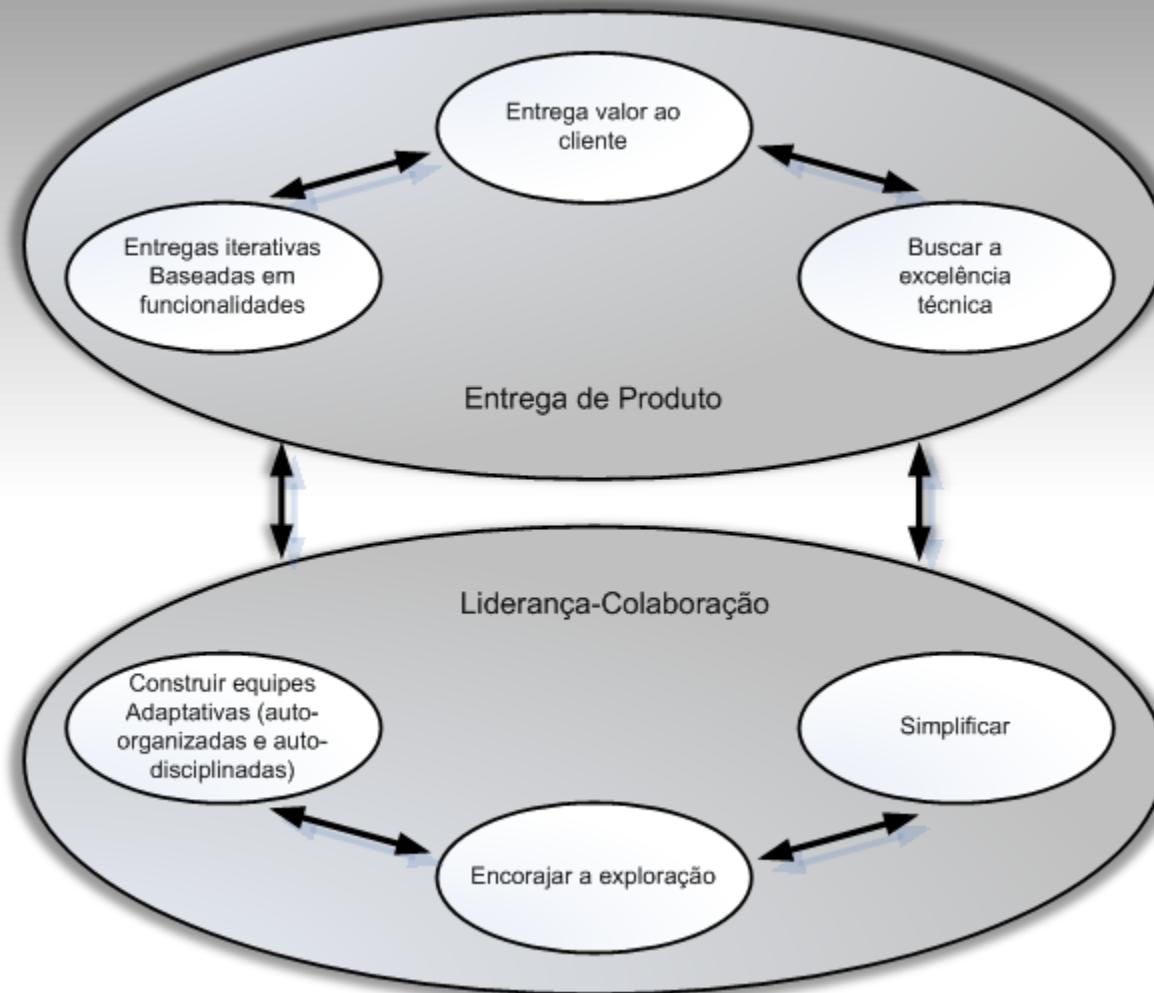
Fonte: HIGHSMITH, 2004

# Visão complementar do ciclo de vida do GAP



Fonte: Sliger & Broderick, 2008

# Princípios do GAP



Fonte: HIGSMITH, 2004

# GP “Tradicional” e GAP

Tópico	Características GP “Tradicional”	Características GAP
Objetivo Principal	Orientado por atividade e centrado em processo.	Orientado por produto e centrado em pessoas.
Tipo de Projeto	Estáveis e com baixo nível de mudanças.	Projetos com mudanças constantes e que necessitam de respostas rápidas.
Tamanho	Aplicável em projetos de todos os tamanhos. Mais efetivo em projetos de maior duração.	Mais efetivo em projetos pequenos (5-10 pessoas) – porém não existe restrições em ser implementado em projetos de maior porte.
Gerente de Projeto	Controle total do projeto.	Papel de facilitador.
Equipe de Projeto	Atuação com papéis claros e bem definidos.	Atuação colaborativa em todas as atividades do projeto.

Fonte: ADAPTADO DE CHIN, 2004

# GP “Tradicional” e GAP

Tópico	Características GP “Tradicional”	Características GAP
Planejamento	Detalhado e os envolvidos têm o papel de validação, não participam da elaboração do planejamento.	Curto e com a participação de todos os envolvidos na elaboração do planejamento.
Arquitetura	Definida com foco em todo o projeto e na reusabilidade	Aplicação de design simples. Evolui junto com o projeto e baseia-se na refatoração.
Modelo de Desenvolvimento	Cascata, espiral e iterativo.	Iterativo e incremental.
Comunicação	Formal.	Informal.
Controle de Mudanças	Processo formal de identificação e aprovação entre os envolvidos. Incorporação de novos requisitos pode ser lento e caro.	Dinâmico e com rapidez de incorporação nas iterações.

Fonte: ADAPTADO DE CHIN, 2004

# Aplicabilidade do GAP (segundo Chin)

	Múltiplas organizações externas interessadas/envolvidas	Múltiplas organizações internas interessadas/envolvidas	Um única organização interessada
Projetos Operacionais	Clássico	Clássico	Clássico
Projetos de Desenvolvimento de novos produtos / processos	Clássico/Ágil	Clássico/Ágil	Ágil
Projetos de desenvolvimento de novas tecnologias / plataformas	Clássico/Ágil	Ágil	Ágil

Clássico: Gerenciamento de Projetos “Tradicional”

Ágil: Gerenciamento Ágil de Projetos

FONTE: CHIN, 2004

# AULA 02

# Metodologias Ágeis de Desenvolvimento de Software

## Extreme Scrum Programming (XP)

Crystal Methods

Adaptive Software Development (ASD)

Lean Development (LD)

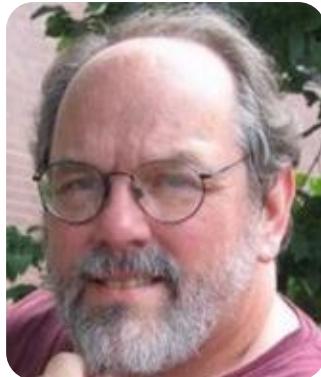
Feature Driven Development (FDD)

Agile Modeling (AM)

Dynamic Systems Development Method (DSDM)

# Extreme Programming – XP (1/5)

- Criado por Kent Beck, Ward Cunningham e Ron Jeffries a partir de suas experiências em um projeto piloto na Daimler Chrysler.



# Extreme Programming – XP (2/5)

- Se baseia em valores e práticas.
- Valores:
  - Feedback
  - Comunicação
  - Simplicidade
  - Coragem

# Extreme Programming – XP (3/5)

- Práticas
  - Cliente Presente
  - Jogo do Planejamento
  - Stand Up Meeting
  - Programação em Par
  - Desenvolvimento Guiado pelos Testes
  - Refactoring
  - Código Coletivo

# Extreme Programming – XP (4/5)

- Práticas
  - Código Padronizado
  - Design Simples
  - Metáfora
  - Ritmo Sustentável
  - Integração Contínua
  - Releases Curtos

# Extreme Programming – XP (5/5)

- Normalmente equipes XP são compostas pelos seguintes papéis:
  - Gerente de Projeto
  - Coach
  - Analista de Teste
  - Redator Técnico
  - Desenvolvedor

# SCRUM



# O que é Scrum? (1/2)

- Scrum é um **framework**, utilizado para organizar times e entregar resultados de maneira mais produtiva e com alta qualidade.

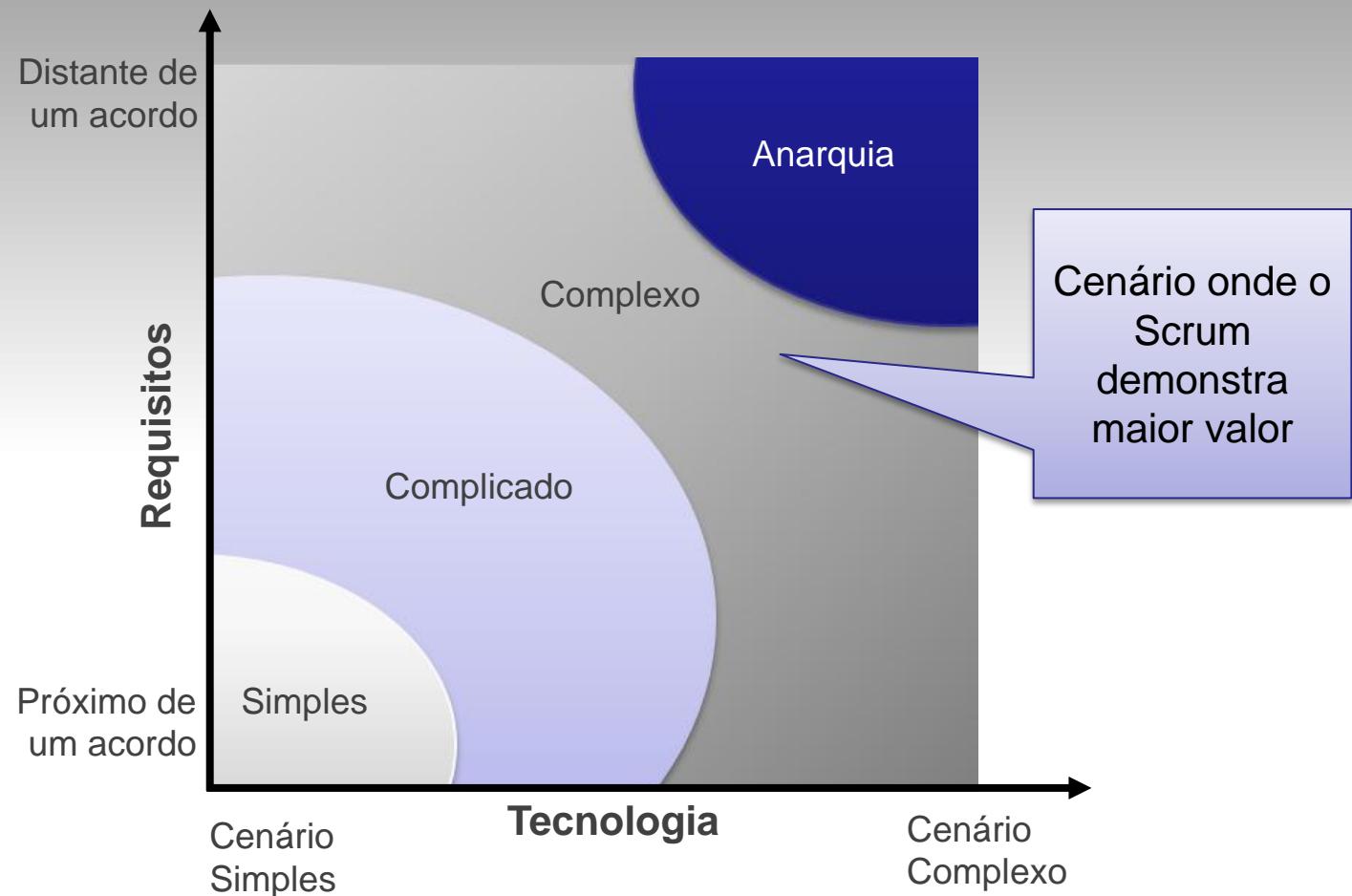
# O que é Scrum? (2/2)

- Fundamentado na teoria de controle de processos empíricos, emprega uma abordagem iterativa e incremental para otimizar a previsibilidade e controlar riscos.

# Scrum não é...

- Uma metodologia que irá fazer desenvolver produtos melhores.
- Uma resposta sobre como desenvolver rapidamente software de qualidade.
- A solução para todos seus problemas...

# Cenário de Complexidade e Scrum



Fonte: ADAPTADO DE SCHWABER, 2008

# Três Pilares do Scrum

TRANSPARÊNCIA

INSPEÇÃO

ADAPTAÇÃO

# Porque Scrum?

- Aumento de ROI
- Flexibilidade
- Produto de Qualidade
- Visibilidade
- Feedback rápido e contínuo

# Raízes do Scrum

- Co-criado por Jeff Sutherland e Ken Schwaber.
- Inspiração inicial a partir do artigo “The New New Product Development Game“ de Hirotaka Takeuchi e Ikujiro Nonaka, publicado em 1986 pela HBR.

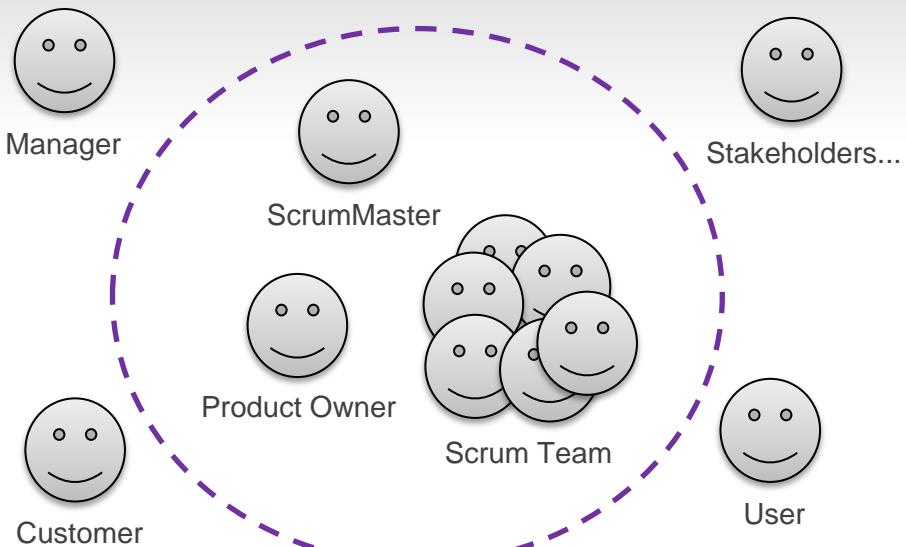


# Framework Scrum

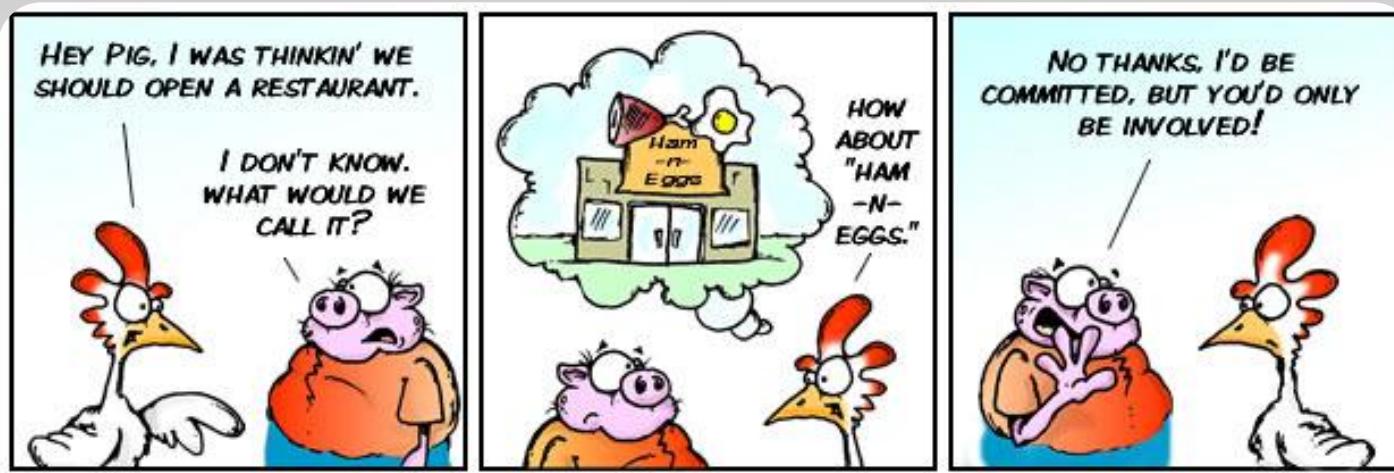
- **Papéis:** Product Owner, ScrumMaster e Time.
- **Cerimônias:** Sprint Planning, Sprint Review e Daily Scrum.
- **Artefatos:** Product Backlog, Sprint Backlog e Burndown chart.

# Papéis

- Product Owner
- ScrumMaster
- Scrum Team



# Porcos e Galinhas



# Product Owner (PO)

- Responsável pelo financiamento do projeto e por maximizar o retorno do investimento (ROI) sob o trabalho que o time Scrum realiza.



# Quem é o seu PO?



# ScrumMaster

- Responsável pelo processo Scrum. Atua como uma espécie de líder e facilitador, trabalha próximo ao PO e deve garantir que o time Scrum esteja operando sob máxima eficiência e eficácia, seguindo as regras e práticas do Scrum.



# Time Scrum

- Responsável pela realização do trabalho.
- Cross-funcional.
- Auto-organizado/auto-contido.
- Tamanho recomendado: 5 a 9 pessoas.

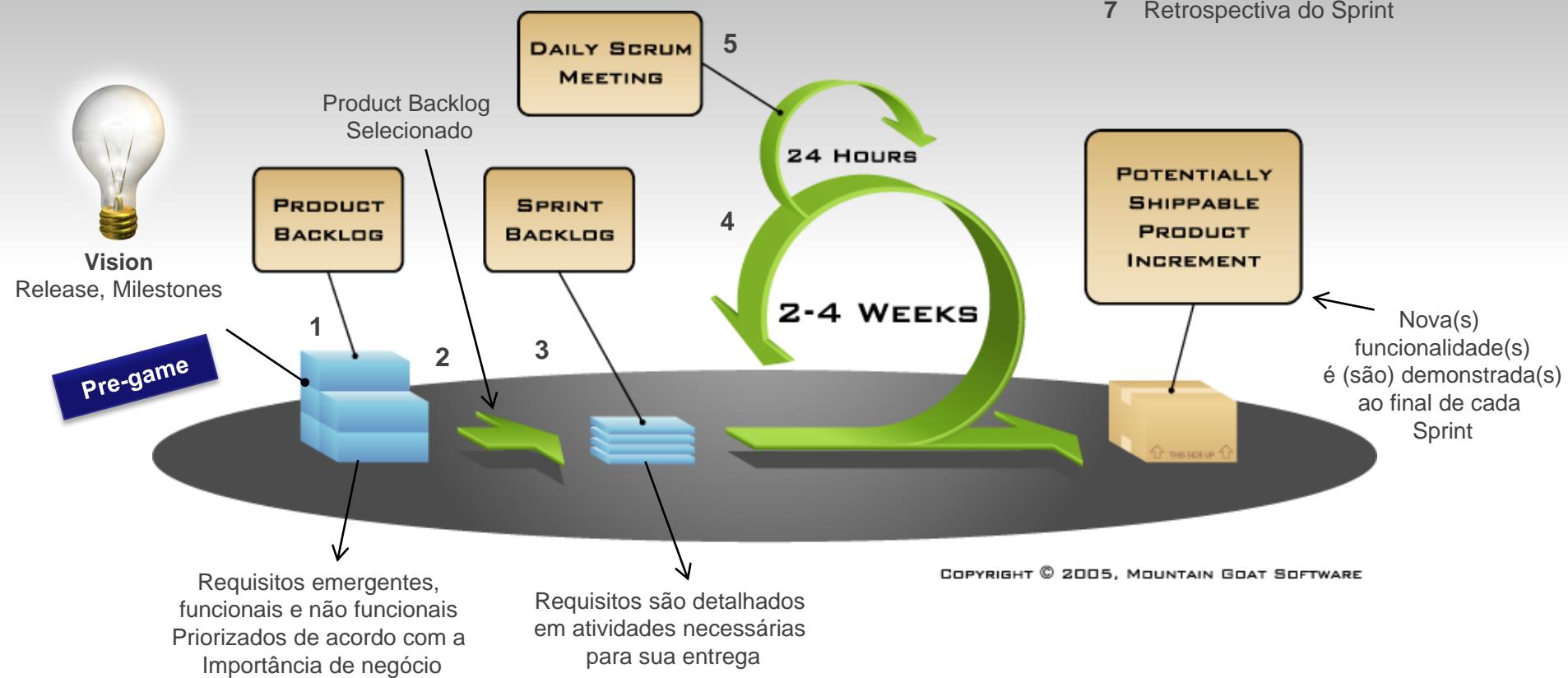


# Time Boxes

- Reunião de Release Planning
- Reunião de Sprint Planning
- Sprint
- Daily Scrum (ou Daily Stand-up)
- Sprint Review
- Reunião de Retrospectiva do Sprint

# Fluxo Scrum

- 1 Reunião de Release Planning
- 2 Reunião de Sprint planning 1 parte
- 3 Reunião de Sprint planning 2 parte
- 4 Sprint
- 5 Daily Scrum
- 6 Sprint Review
- 7 Retrospectiva do Sprint



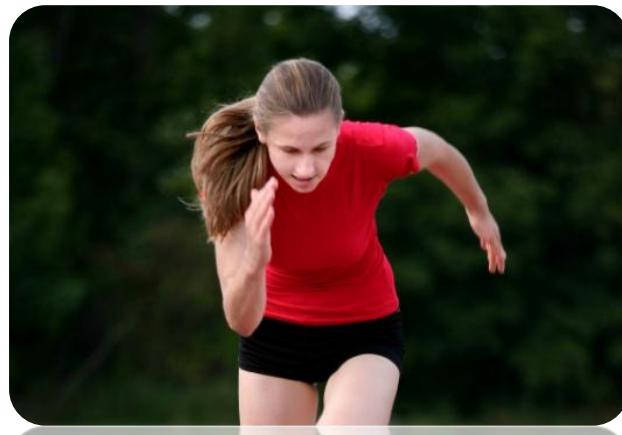
# Reunião de Release Planning

- Estabelece a meta da release e define as maiores prioridades do Product Backlog.
- Analisar os principais riscos.
- As características gerais e funcionalidades que estarão contidas na release.
- Estabelece uma data de entrega e custo prováveis.



# Sprint

- É uma iteração (não interação)
- Time-boxed
- Todo trabalho é realizado nos Sprints
- Consiste no Sprint Planning, desenvolvimento do trabalho, Sprint Review e Restrospectiva.



# Sprint – Interrupção Anormal

- Pode ser realizada caso o PO determine que não faz sentido terminar o trabalho iniciado.
- Este tipo de interrupção deve ser evitada sempre que possível.
- Começa-se um novo Sprint.



# Reunião de Planejamento do Sprint

- Momento em que planeja-se os detalhes do Sprint.
- Usualmente requer 1 dia de trabalho (8 horas).
- Divido em 2 momentos.



# Daily Scrum (stand-up)

- 15 minutos (não mais)
  - O que você fizeste ontem?
  - O que irá fazer hoje?
  - Existe algum obstáculo em seu caminho?



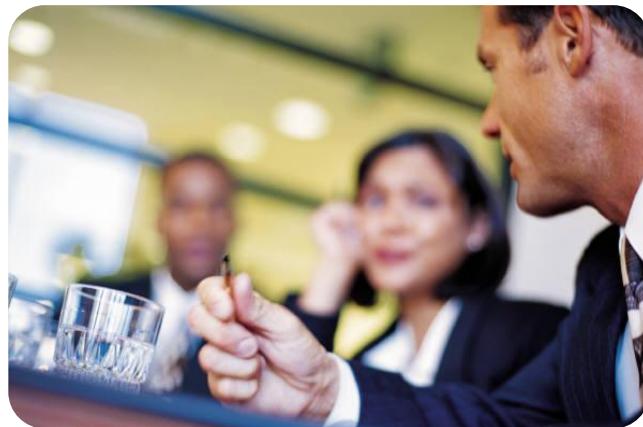
# Reunião de Revisão do Sprint

- O trabalho/funcionalidades construído durante o Sprint é apresentado ao Product Owner, qual avalia a aceitação ou não das entregas.
- Momento de inspeção e adaptação.



# Retrospectiva da Sprint

- Quais foram os pontos positivos da Sprint?
- O que podemos melhorar para a próxima Sprint?
- ScrumMaster é o facilitador. Participação do PO é opcional. Reunião deve resultar em uma lista de ações a serem endereçadas no próximo Sprint.



# Artefatos

- Product Backlog
- Sprint Backlog
- Release Burndown
- Sprint Burndow

# Product Backlog

- A “lista de desejos” do PO.
- Itens priorizados de acordo com seu valor / importância de negócio.
- PO é o dono deste artefato.
- Enquanto o produto existir, o Product Backlog também exisitirá.

	Item #	Description	Est	By
<b>Very High</b>				
	1	<b>Finish database versioning</b>	16	KH
	2	<b>Get rid of unneeded shared Java in database</b>	8	KH
	-	<b>Add licensing</b>	-	-
	3	Concurrent user licensing	16	TG
	4	Demo / Eval licensing	16	TG
	<b>Analysis Manager</b>			
	5	File formats we support are out of date	160	TG
	6	Round-trip Analyses	250	MC
<b>High</b>				
	-	<b>Enforce unique names</b>	-	-
	7	In main application	24	KH
	8	In import	24	AM
	-	<b>Admin Program</b>	-	-
	9	Delete users	4	JM
	-	<b>Analysis Manager</b>	-	-
	10	When items are removed from an analysis, they should show up again in the pick list in lower 1/2 of the analysis tab	8	TG
	-	<b>Query</b>	-	-
	11	Support for wildcards when searching	16	T&A
	12	Sorting of number attributes to handle negative numbers	16	T&A
	13	Horizontal scrolling	12	T&A
	-	<b>Population Genetics</b>	-	-
	14	Frequency Manager	400	T&M
	15	Query Tool	400	T&M
	16	Additional Editors (which ones)	240	T&M
	17	Study Variable Manager	240	T&M
	18	Haplotypes	320	T&M
	19	<b>Add icons for v1.1 or 2.0</b>	-	-
	-	<b>Pedigree Manager</b>	-	-
	20	Validate Derived kindred	4	KH
<b>Medium</b>				
	-	<b>Explorer</b>	-	-
	21	Launch tab synchronization (only show queries/analyses for logged in users)	8	T&A
	22	Delete settings (?)	4	T&A

Fonte: <http://epf.eclipse.org/wikis/scrum/>

# Sprint Backlog

- Detalha as atividades necessárias para transformar os itens selecionados do Product Backlog em um incremento do Sprint potencialmente entregável (pronto).

Tasks	Mon	Tues	Wed	Thurs	Fri
Code the user interface	8	4	8		
Code the middle tier	16	12	10	4	
Test the middle tier	8	16	16	11	8
Write online help	12				
Write the foo class	8	8	8	8	8
Add error logging			8	4	

Fonte: <http://epf.eclipse.org/wikis/scrum/>

# Sprint Burndown

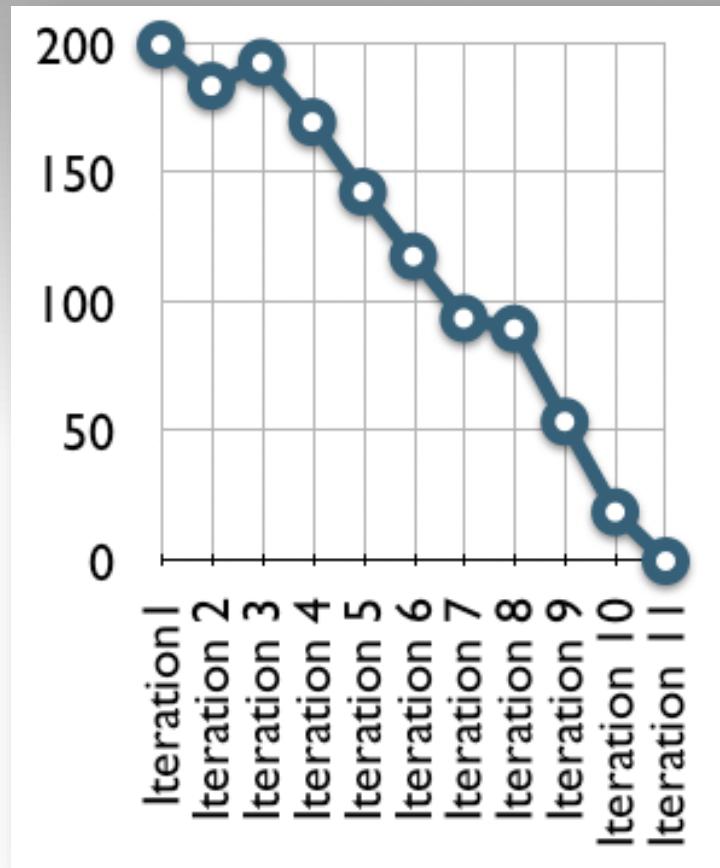
- Demonstra o progresso da Sprint.
- Quanto do backlog de atividades necessárias o time já “queimou” no Sprint.



Fonte: <http://epf.eclipse.org/wikis/scrum/>

# Release Burndown

- Demonstra o progresso da Release.
- Quanto do Product Backlog priorizado o time já entregou.
- Atualizado ao final de cada Sprint.
- Unidade de medida pode ser *Story Points* entregues em cada Sprint (não é regra).



Fonte: <http://epf.eclipse.org/wikis/scrum/>

# Definição de Pronto (*DoD*)

- Scrum exige que os times desenvolvam um incremento de funcionalidade do produto a cada Sprint.
- Time deve ter uma clara definição do conceito de “Pronto” junto ao PO.
- O que é “Pronto” para você?



# Exemplo de uma lista de “Pronto”

DoD Checklist:

- Desenvolver a funcionalidade
- Testar unitariamente
- Testar a integração com outros componentes (quando for o caso)
- Verificar se o build do projeto funciona sem erros e fazer o deploy em uma ambiente de produção simulado
- Testar segundo os critérios de aceitação estabelecidos pelo cliente
- Depois dos testes desenvolvidos e a nova funcionalidade passando em todos eles, avaliar a necessidade de fazer refactoring no novo código
- Com a entrada da nova funcionalidade, avaliar a necessidade de fazer refactoring em algum módulo do sistema
- Atualizar a documentação (quando necessário)

# Pronto



By Clark & Vizzini

© 2006 implementingscrum.com

© 2006 implementingscrum.com

© 2006 implementingscrum.com

# Escalando Scrum

- Scrum em ordem primária funciona de maneira mais efetiva em times pequenos e localizados em um mesmo ambiente.
- Porém existem práticas para escalar Scrum a mais de um time, mesmo estando estes descentralizados (ex.: Scrum of Scrums).
- Processo de escalar Scrum deve ser planejado de maneira atenta aos possíveis problemas de distância e aumento dos canais de comunicação.

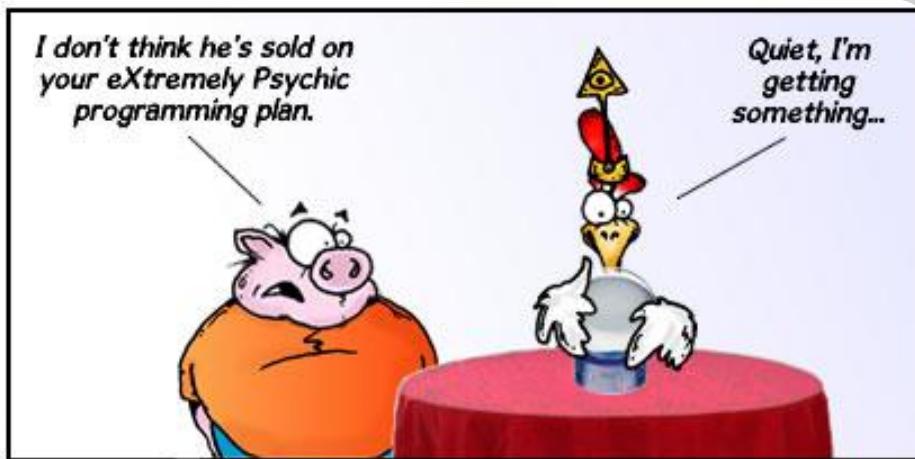
# AULA 03

# **VISÃO GERAL DE PLANEJAMENTO ÁGIL (PLANEJANDO COM SCRUM)**

# Scrum não planeja?



By Clark & Vizdos



© 2007 implementingscrum.com

© 2001 implementingscrum.com

# O que faz um planejamento ser Ágil?

- É mais focado no processo contínuo de planejamento do que em um plano.
- Encoraja a mudança.
- Resulta em planos fáceis de serem modificados.
- É utilizado de maneira ativa durante todo o projeto.

Fonte: COHN, 2006

# Diferentes níveis de planejamento



Fonte: COHN, 2006

# Planejamento em diferentes níveis

Visão do Produto (longo prazo, estratégica)



Roadmap do Produto (release 1, release 2...)

Épicos

Plano de Release (sprint 1, sprint 2...)

Temas

Plano de Iteração(story. x, y...)

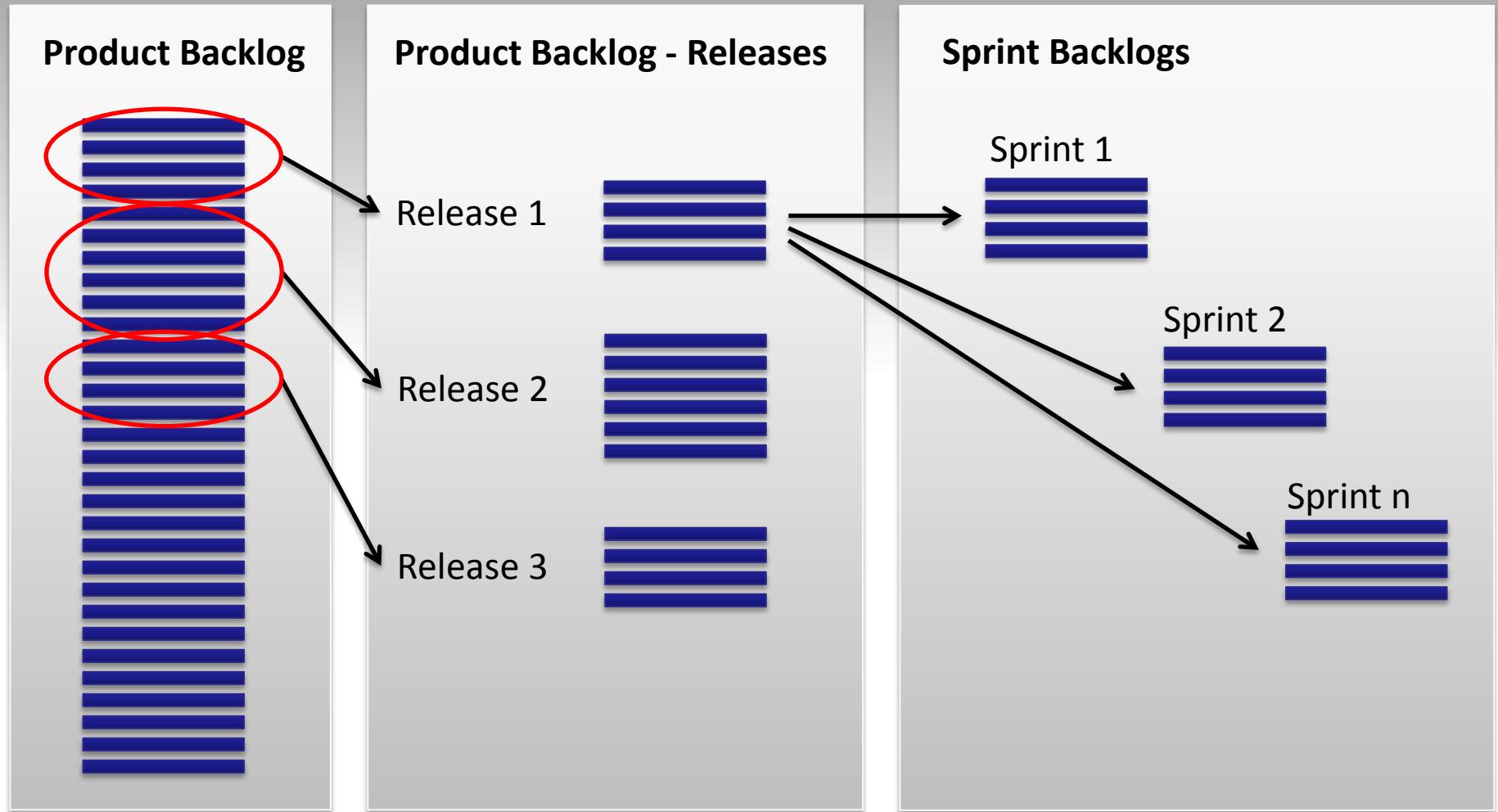
User Stories

Trabalho diário  
(atividade x, y...)

Atividades



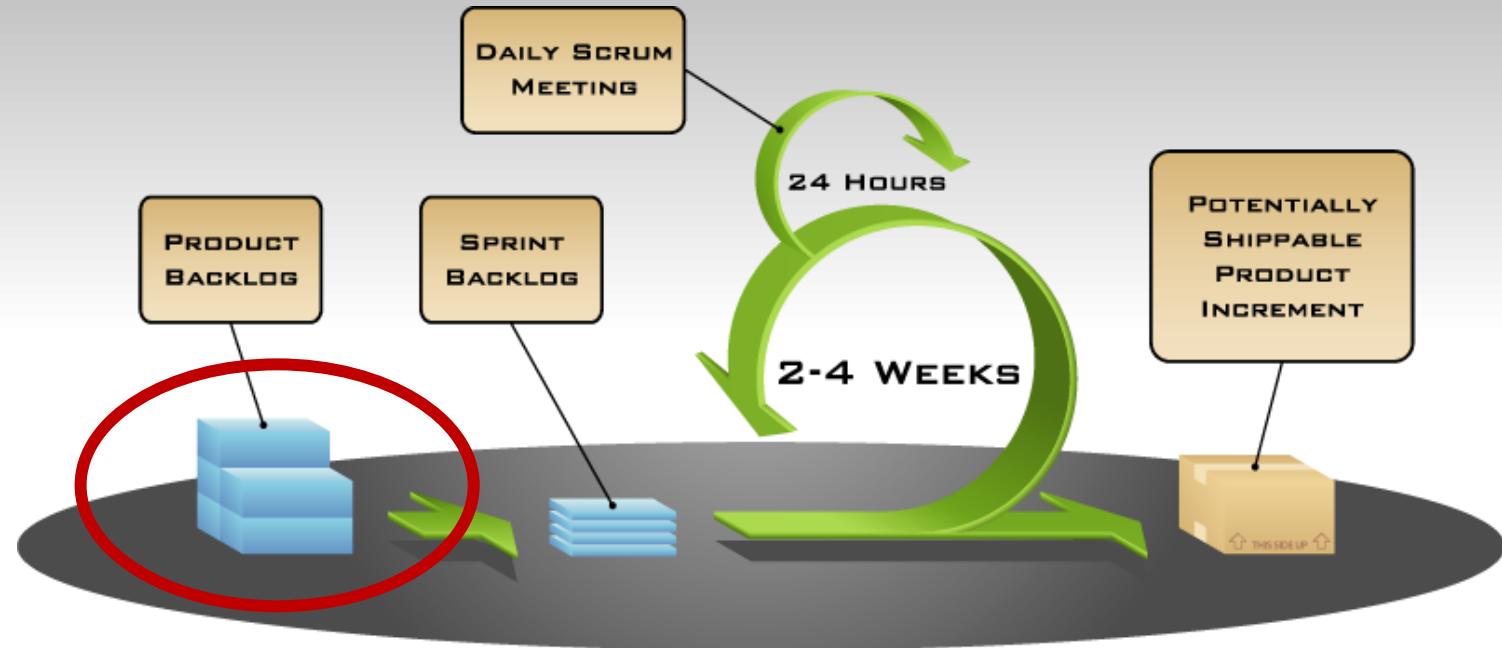
# Product Backlog ao Sprint Backlog



# Plano de Release

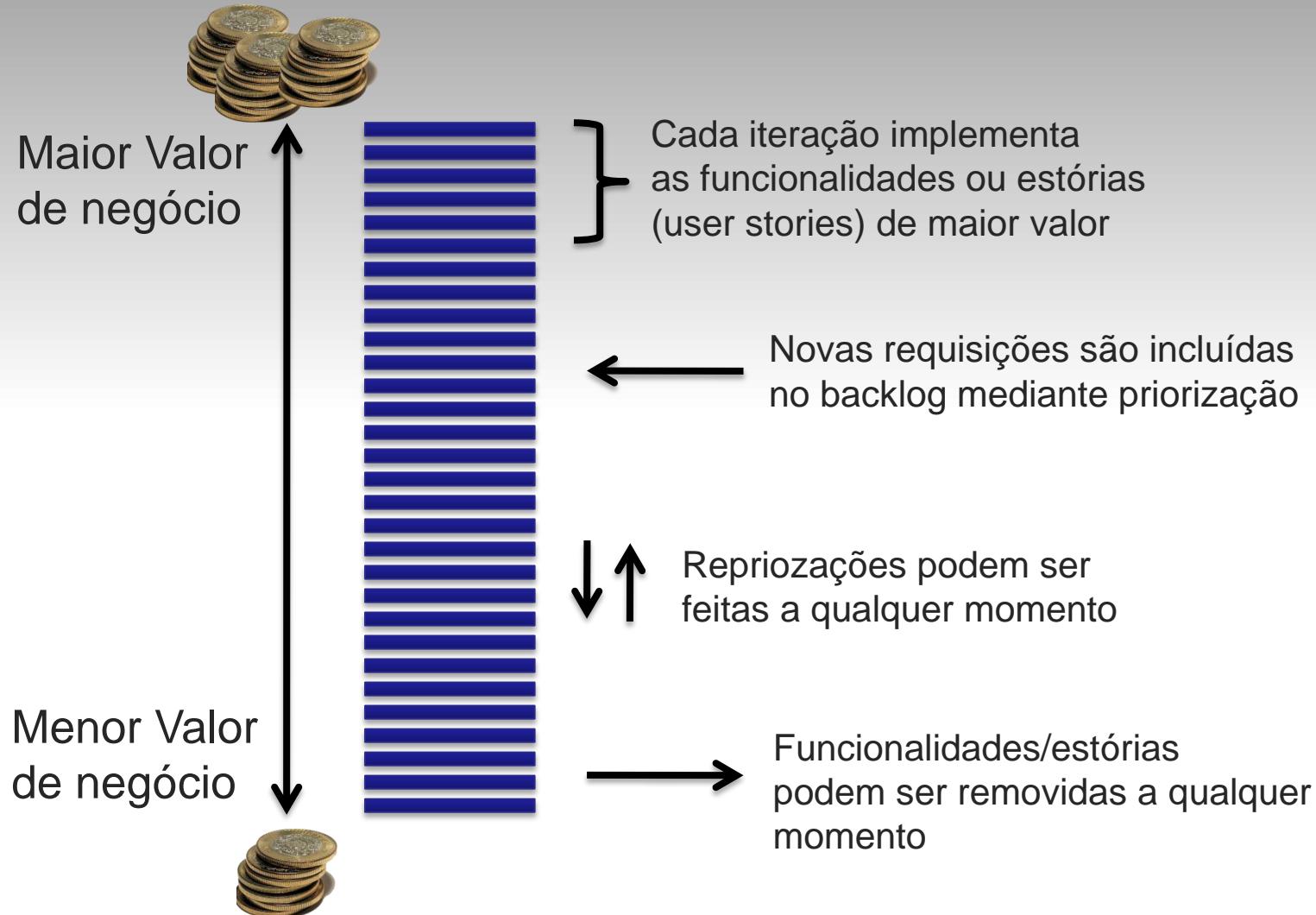
Sprint 1	Sprint 2	Sprint 3 - 5		
História x	História a	História c	História c	História a
História y	História b	História d	História d	História b
História z		História e	História e	
Atividade	Horas	História f		
Atividade A	8			
Atividade B	4			
Atividade C	8			

# Product Backlog

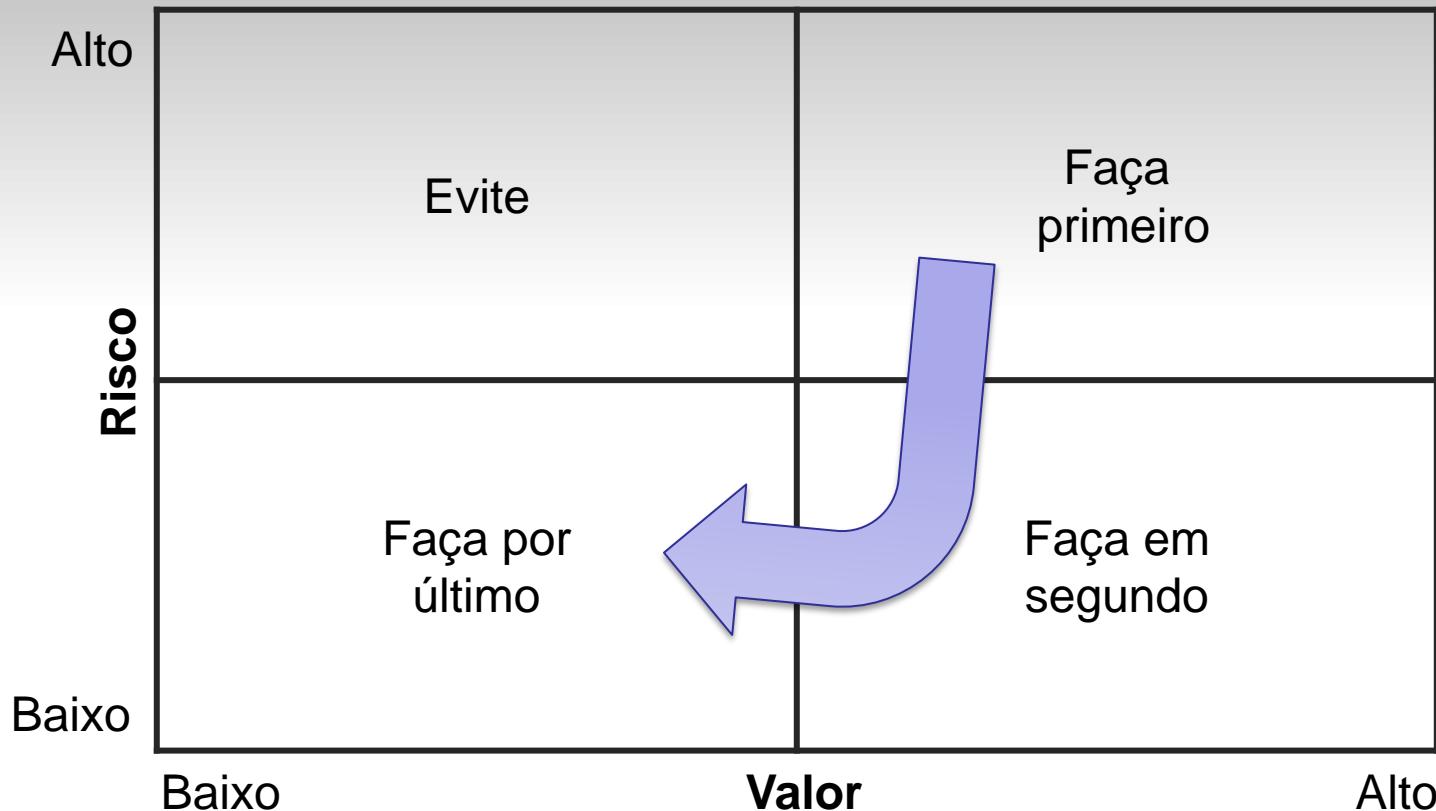


COPYRIGHT © 2005, MOUNTAIN GOAT SOFTWARE

# Priorização do Product Backlog

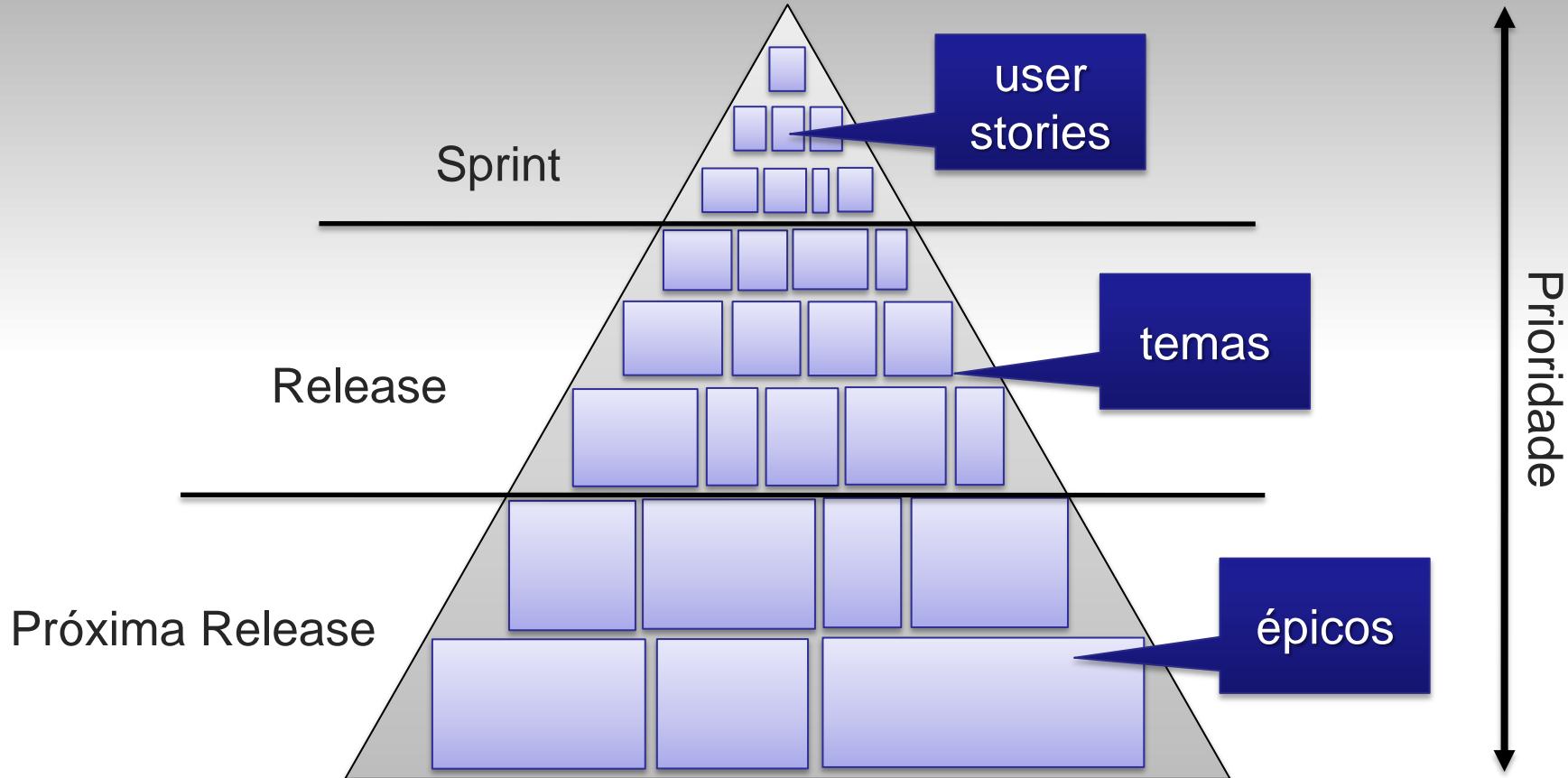


# Combinando Risco e Valor para priorizar o Backlog



Fonte: COHN, 2006

# Product Backlog Iceberg



# User Stories (Histórias do Usuário)

- É uma parte da funcionalidade do sistema, entendido pelo cliente / Product Owner, que representa um incremento de valor de negócio a ser implementado pela equipe.
- Tradicionalmente escritas em cartões.
- Meio de comunicação entre time e cliente sobre os requisitos a serem desenvolvidos.

# *User Stories (Histórias do Usuário)*

- 3 C's de Ron Jeffries':
  - Cartão
  - Conversação
  - Confirmação

Como um <ator>,  
eu gostaria de <ação>,  
para <motivo/valor de negócio>.

Como um usuário do LinkedIn,  
eu quero atualizar o meu perfil  
para que eu possa manter minhas  
informações atualizadas.

Como sistema A, eu quero  
acompanhar a capacidade do  
Sistema B, de modo que eu possa  
emitir alertas apropriados  
quando o mesmo atingir um  
certo limite.

Como um usuário do LinkedIn, eu quero atualizar o meu perfil para que eu possa manter minhas informações atualizadas.



Prática comum: escrever condições de satisfação da *User Story* atrás dos cartões como forma de estender o entendimento da funcionalidade, suas restrições, etc. junto ao cliente

Como demonstrar:

- [ ] Verifique se os campos de preenchimento obrigatório estão preenchidos.
- [ ] Verifique se o usuário recebe confirmação de atualização do perfil.
- [ ] Etc.

# Tamanho Ideal para *User Stories*

- Nem tão grande que não possa ser específico e dividida em partes menores (Épicos) .
- Nem tão pequena que não seja facilmente entendida pelo cliente/Product Owner e perca seu significado de negócio (decomposição da User Story a um nível que possa confundir-se com as atividades/tarefas).

# Uma *User Story* deve ser...

- Independente
- Negociável
- Valiosa para o cliente/PO ou usuário
- Estimável
- Pequena (mantendo valor de negócio)
- Testável

# **ESTIMATIVAS**

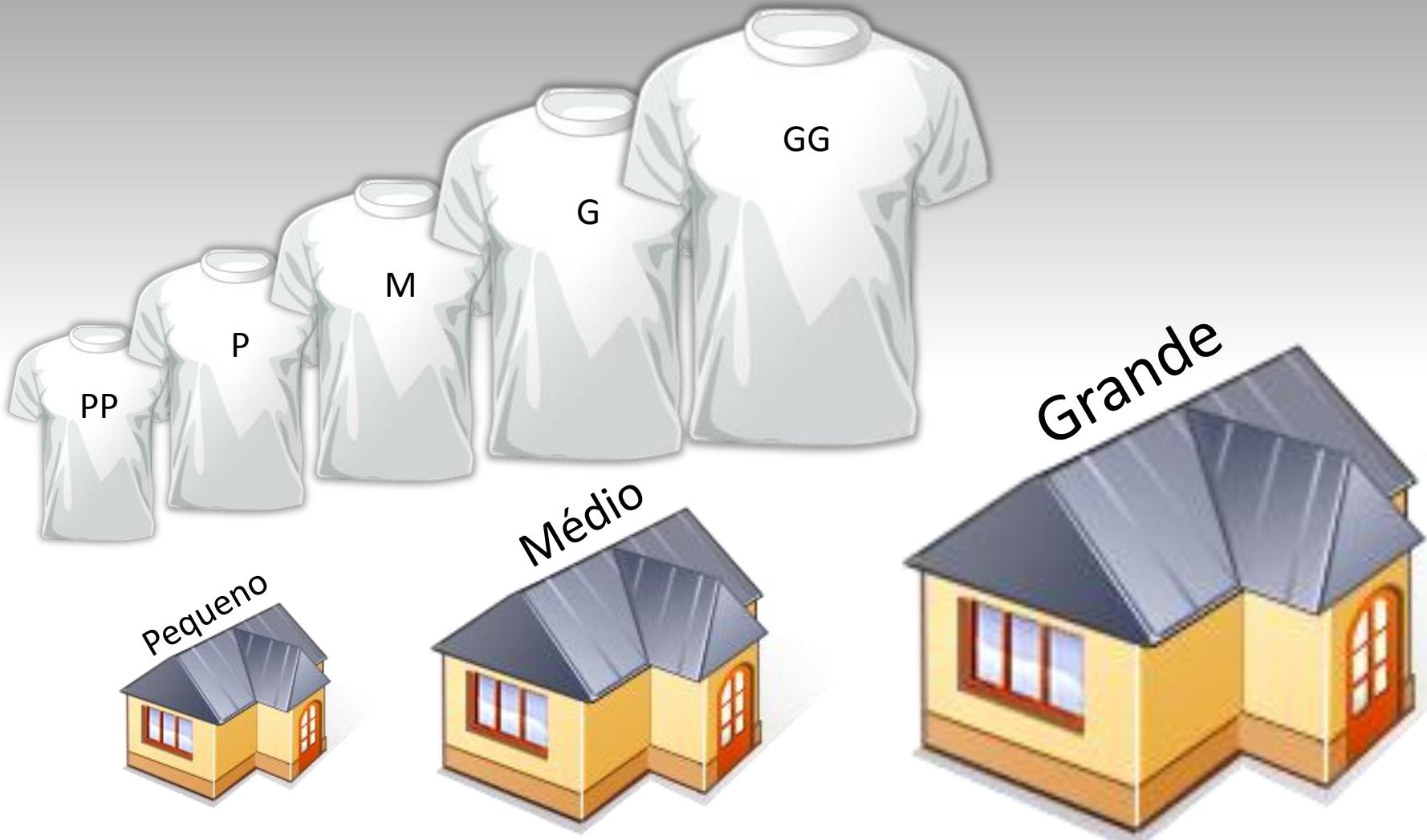
# Estimativas

- Razões para estimar:
  - Previsão
  - Performance
  - Priorização
- Erro comum ao estimar:
  - Transformar estimativas em compromissos

# *Story Points*

- Mede tamanho relativo, não tempo.
- Usualmente utiliza-se a escala Fibonacci (1,2,3,5,8,13,20,40,100).
- Atribuição do valor deve ser feita pelo time e não por uma pessoa.

# Estimando Tamanho



# Tempo Ideal (*Ideal Days*)

- Pense em seu dia de trabalho sem interrupção alguma.

Dia de trabalho (~8 horas)

- (-) Reuniões
- (-) Conversas no café
- (-) Almoço
- (-) Interrupções do colega/chefe/etc.
- (-) Leitura de notícias na web/etc.

---

= *Ideal Day*

# *Story Points ou Ideal Days?*

- Resposta: depende da sua realidade e entendimento do time sobre estas unidades de medida.
- Comumente *Ideal Days* tende a ser melhor aceito em times que estão iniciando seu caminho em abordagem ágil.

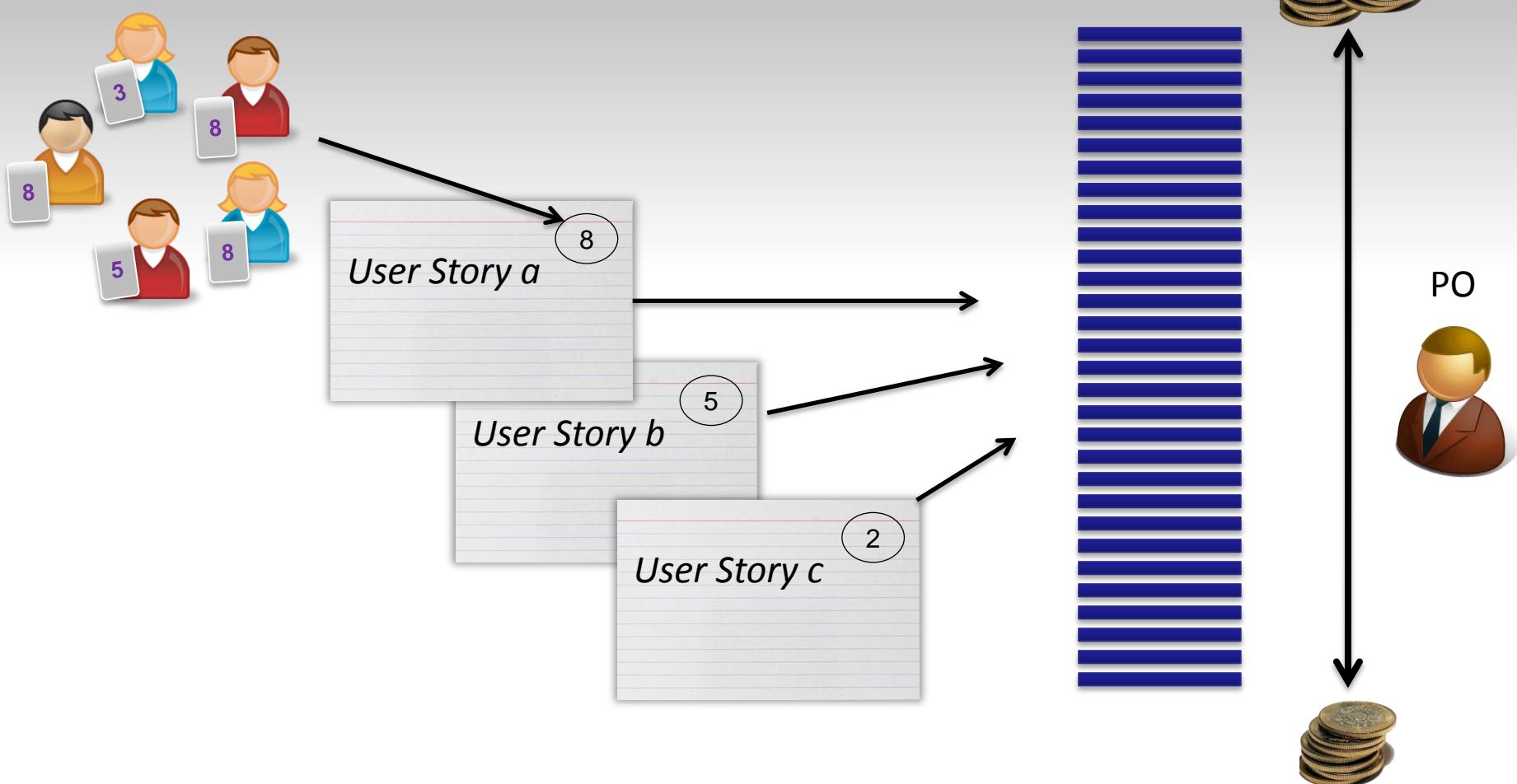
# *Planning Poker (1/2)*

- Técnica de estimativa que busca o consenso.
- Primeiramente descrita por James Grenning em 2002 e popularizada por Mike Cohn em seu livro *Agile Planning and Estimating*.
- Apesar de usualmente ser utilizada no processo de planejamento ágil, não trata-se da única técnica (utiliza-se outras como opinião de especialista, analogia, etc.)

# *Planning Poker (2/2)*



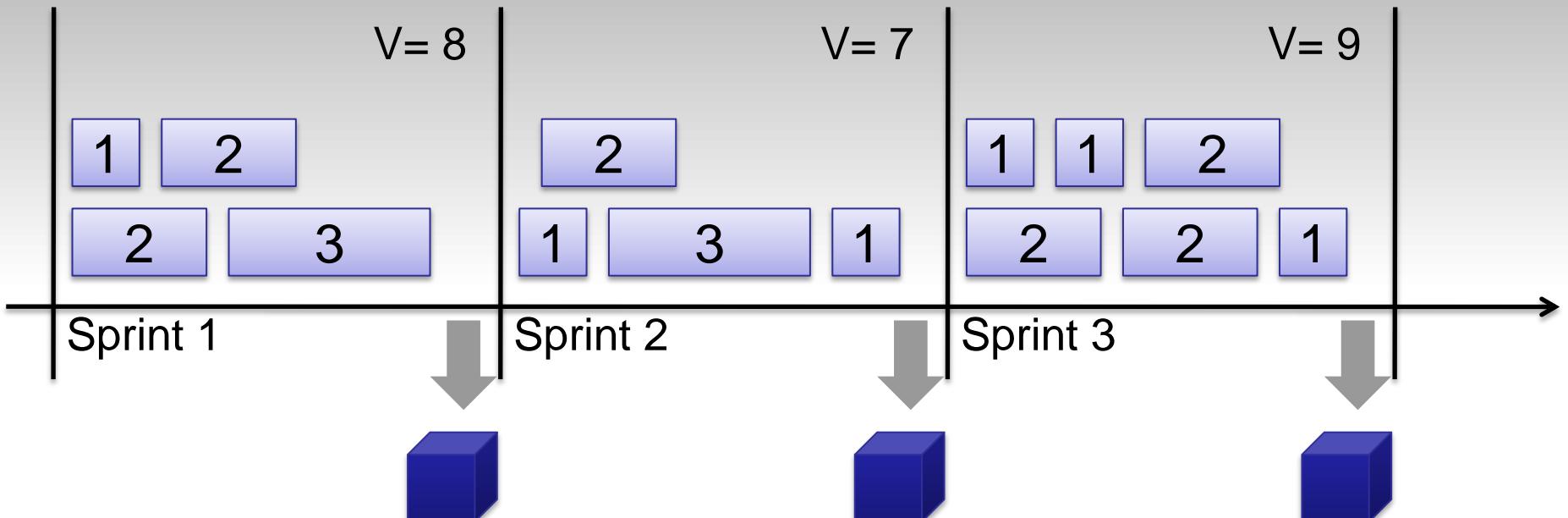
# Voltando ao Product Backlog por 1 min...



# *Velocity (1/2)*

- A velocidade com a qual um time consegue converter itens do Product Backlog em produto funcional.
- Um observação empírica da capacidade de um time completar determinado conjunto de trabalho por iterações.
- Em um cenário de utilização de *Story Points* refere-se a quantidade de “pontos” entregues ao final de uma Iteração/Sprint.

# Velocity (2/2)

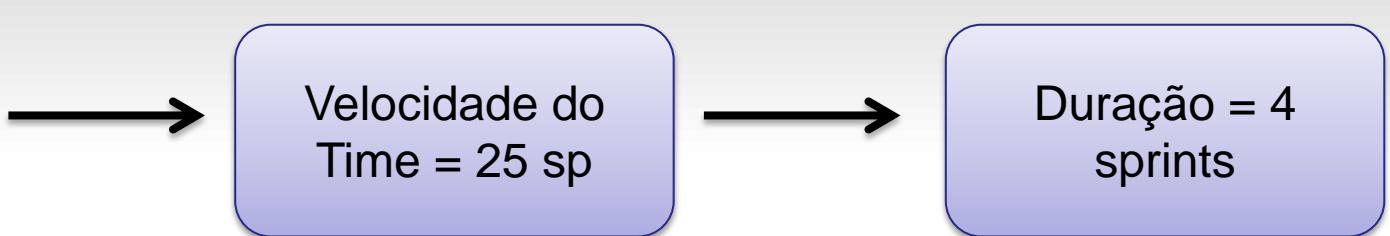
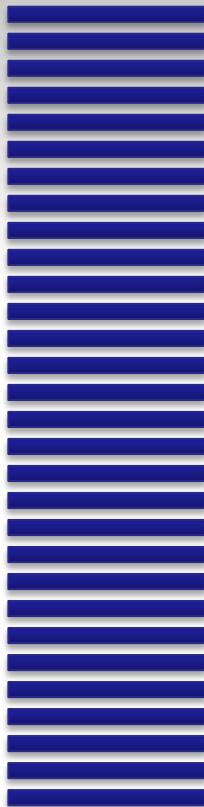


Estimativa de Velocity: 7-9 pontos por Sprint

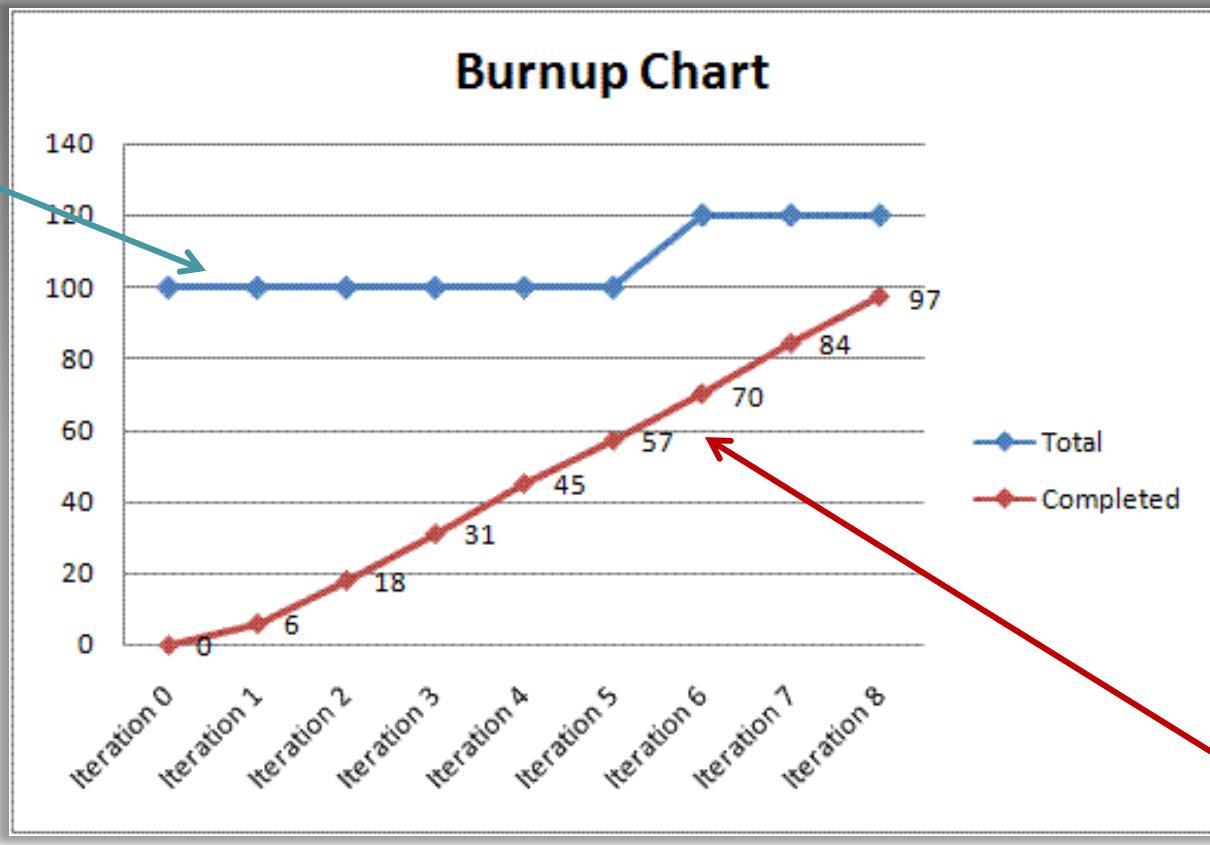
Fonte: <http://crisp.se/henrik.kniberg/>

# Estimativa / Velocity = Duração

Product Backlog  
da Release = 100 *story points*



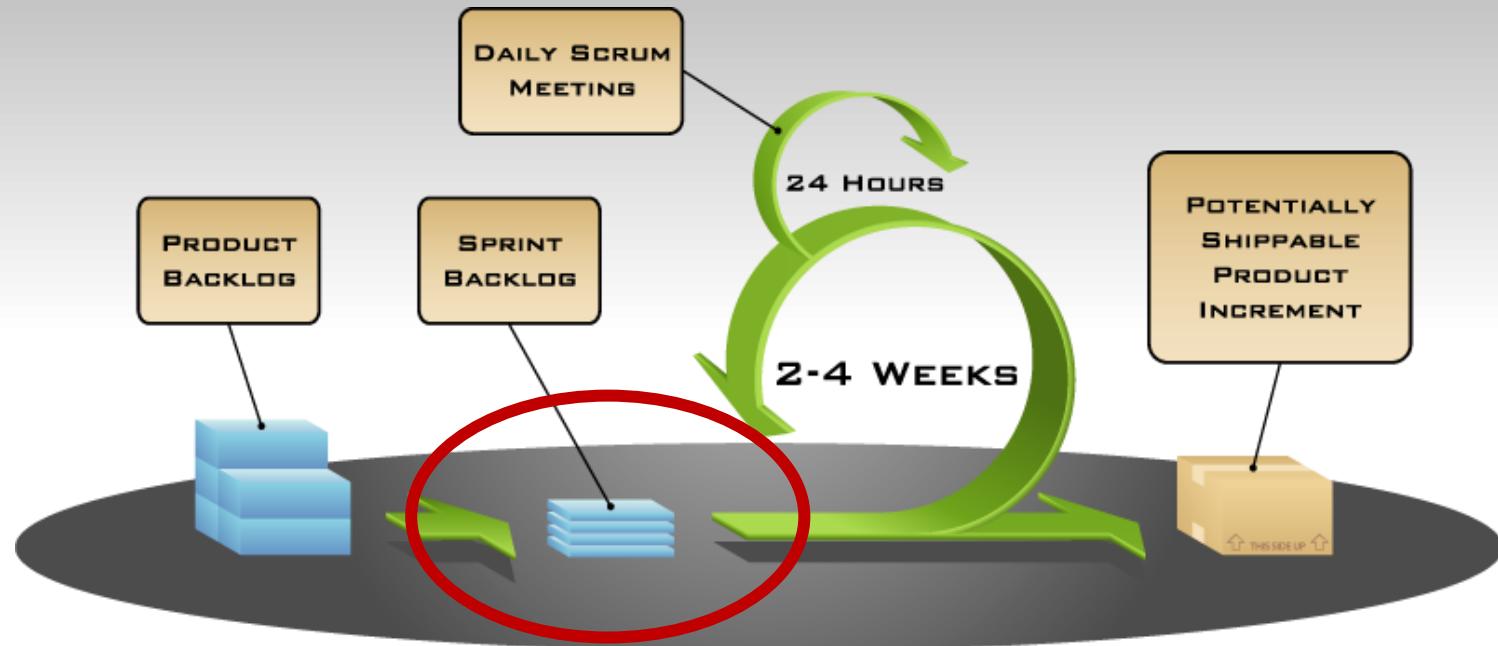
# Gráfico Burnup



Escopo

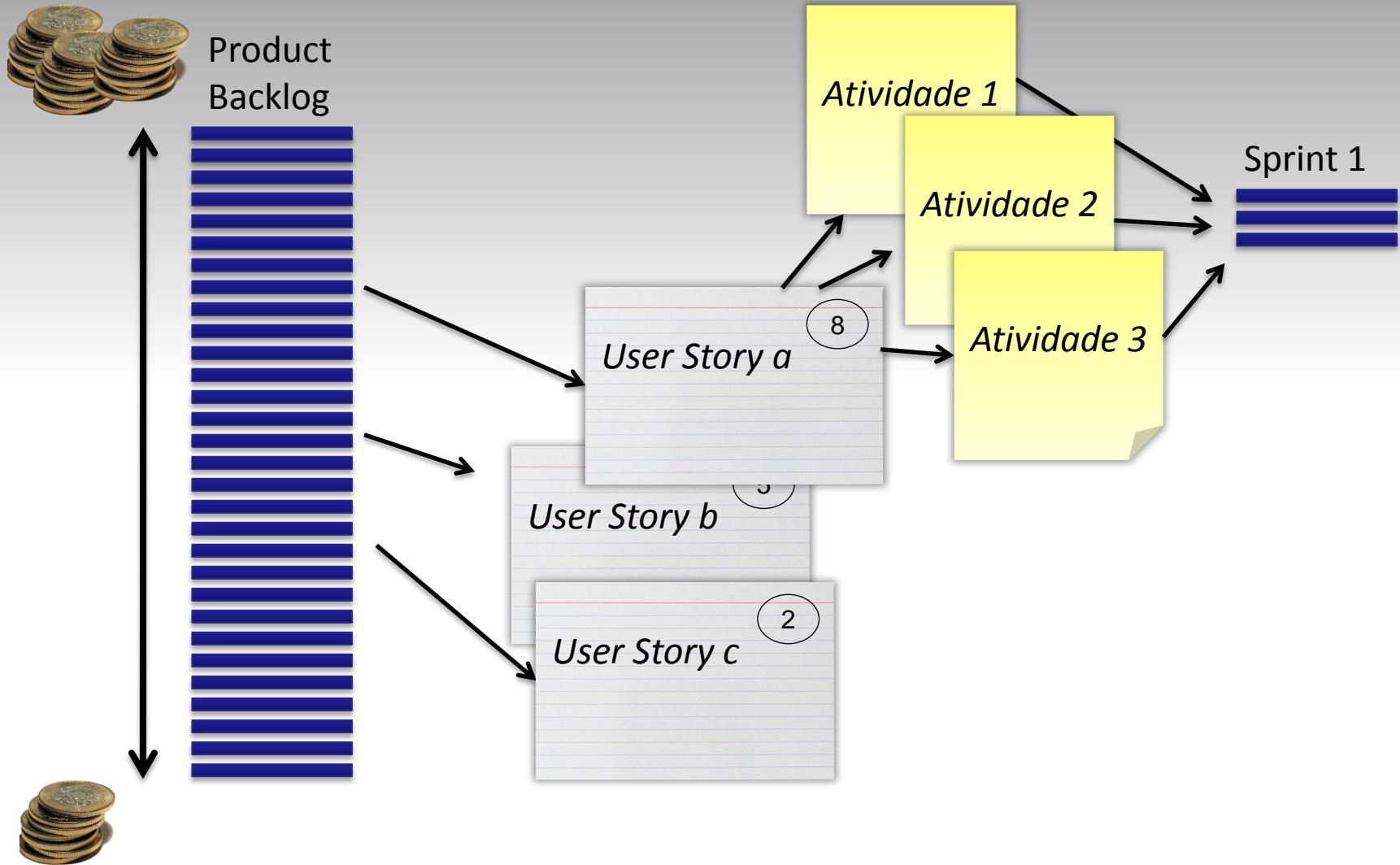
Progresso

# Sprint Backlog



COPYRIGHT © 2005, MOUNTAIN GOAT SOFTWARE

# Do Product Backlog ao Sprint Backlog



# Task Board

Story	To Do	In Process	To Verify	Done
As a user, I... 8 points	Code the... 9 Code the... 2 Test the... 8	Test the... 8 Code the... 8 Test the... SC 8	Code the... DC 4 Test the... SC 6	Code the... DC 8 Test the... SC 8 Test the... SC 8 Test the... SC 6
As a user, I... 5 points	Code the... 8 Code the... 4	Test the... 8 Code the... 6	Code the... DC 8	Test the... SC 8 Test the... SC 6 Test the... SC 6

Fonte: <http://epf.eclipse.org/wikis/scrum/>

# AULA 04

# **BREVE CONEXÃO DO GP “TRADICIONAL” A ABORDAGEM ÁGIL**

# PMBOK e Agilidade

- O PMBOK é um grande e organizado compilado de conhecimentos relacionados a melhores práticas, técnicas e teorias de gestão ligadas ao gerenciamento de projetos.
- Não existe algo como PMBOK vs Gerenciamento Ágil de Projetos...

# Gerenciamento da Integração

## GP “Tradicional”

Desenvolver o plano de gerenciamento do projeto

Execução do Plano de Projeto

Monitorar e controlar o trabalho do projeto

Realizar o controle integrado das mudanças

## GAP

Planejamento da release e iterações

Trabalho Iterativo

Facilitar, inspecionar, adaptar, liderar, colaborar

Feedback constante e Backlog priorizado

Não exaustivo em relação a todos processos do PMBOK

# Gerenciamento do Escopo

## GP “Tradicional”

Definir o escopo

Criar WBS (EAP)

Verificar o escopo

Controlar o escopo

## GAP

Backlog e reuniões de planejamento

Criar FBS

Definição dos critérios de aceitação e envolvimento do cliente

Feedback constante e Backlog priorizado

Não exaustivo em relação a todos processos do PMBOK

# Gerenciamento do Tempo

## GP “Tradicional”

Definir as atividades

Sequenciar as atividades

Estimar os recursos das atividades

Desenvolver o cronograma

Controlar o cronograma

## GAP

Planejamento da Iteração

Planejamento da Iteração

Membros do time escolhem e estimam as atividades

Roadmap do projeto, planos de release e iteração

Atualizar roadmap e planos de release com base na velocity

Não exaustivo em relação a todos processos do PMBOK

# Gerenciamento dos Custos

## GP “Tradicional”

Estimar os custos

Determinar o orçamento

Controlar os custos

## GAP

Plano de Release

Orçamento de  
Release/Iteração

Priorizar o Product  
Backlog

Não exaustivo em relação a todos processos do PMBOK

# Gerenciamento da Qualidade

## GP “Tradicional”

Planejar a qualidade

Realizar a garantia da  
qualidade

Realizar o controle da  
qualidade

## GAP

Critérios de Aceitação e  
definição de pronto

Qualidade como parte  
intrínseca das atividades  
do time

Definição dos critérios de  
aceitação / pronto; Testar  
cedo e frequentemente

Não exaustivo em relação a todos processos do PMBOK

# Gerenciamento dos Riscos

## GP “Tradicional”

Identificar os riscos;  
Análise Qualitativa;  
Planejar as respostas aos  
riscos

Monitorar e controlar os  
riscos

## GAP

Planejamento Iterativo;  
Daily Stand-ups e  
Retrospectivas

Stand-ups e radiadores  
de informação altamente  
visíveis

Não exaustivo em relação a todos processos do PMBOK

# **CONTRATOS NA ABORDAGEM ÁGIL**

# Contratos no contexto Ágil

- Contrato deve preocupar-se em detalhar a forma de interação/comunicação e trabalho entre as partes.
- Menos preditivo no que tange a definição detalhada do produto final (apresentar visão a qual se busca alcançar).
- Relação de confiança entre as partes é fundamental.
- Disponibilidade de usuários de negócio para responder as questões do projeto deve ser um requisito contratual.

# Contratos no contexto Ágil

- Contratos do tipo preço fixo não são indicados (preço fixo = escopo fixo).
- Contratos do tipo *Time & Material* são os mais indicados.
- Faturamento preferencialmente alinhado com as iterações/sprints.
- Sign-off do cliente ao final de cada iteração ajuda no controle de progresso do Contrato.

# **GERENTE DE PROJETOS E O LÍDER NO CONTEXTO ÁGIL**

# Papel do Gerente de Projetos

- Reconhecer que projetos ágeis irão mudar de direção muitas vezes durante seu curso.
- Mapear as influências externas que possam impactar o projeto e compartilhar de maneira adequada essa informação com os times.
- Atuar como um “canalizador” de informações, provendo informações de valor para o time de projeto.
- Manter uma visão geral do projeto para o time.

# Papel do Gerente de Projetos

- Facilitar o processo de interação e comunicação entre as pessoas.
- Liderar o processo de adaptação através da manutenção das lições aprendidas, atuando nas ações corretivas.
- Proteger o time das distrações externas.
- Criar um ambiente seguro, qual incentiva a colaboração, o tomada de decisão e encoraja a experimentação.
- Manter um ambiente que suporta a alta produtividade.

# **CONSIDERAÇÕES FINAIS**

# Considerações finais (1/2)

- Lembrando: metodologias ágeis não são a solução para todos os problemas (identifique o cenário/contexto).
- Mudar o modelo mental de pensar o planejamento/desenvolvimento de projetos de software.
- Processo de mudança deve envolver alta diretoria (arranje o seu SPONSOR).
- Lembre sempre dos valores e princípios.



# Considerações finais (2/2)

- Scrum é simples na teoria e difícil na prática.
- Para uma implantação bem sucedida é vital aliar as práticas ágeis de gestão com práticas técnicas (TDD, Refatoração, Pair Programming, Integração Contínua, etc.) - envolva sua equipe!
- Mudança requer coragem e persistência.
- Foco na melhoria contínua.



# FIM

# Referências citadas

- AGILLE ALLIANCE. *Manifesto for agile software development*. Disponível em <<http://www.agilemanifesto.org>>.
- BOEHM, B. W. *A Spiral Model of Software Development and Enhancement*. Computer, v.21, n.5, p.61-72, 1988.
- COHN, M. Agile Estimating and Planning. NJ: Prentice Hall, 2006.
- CHIN, G. *Agile Project Management: how to succeed in the face of changing project requirements*. NY: Amacon, 2004.
- HIGHSMITH, J. *Agile Software Development Ecosystems*. Boston: Addison-Wesley, 2002.
- HIGHSMITH, J. *Agile Project Management: creating innovative products*. Boston: Addison-Wesley, 2004.
- PRESSMAN, R.S. *Engenharia de Software*. 6.ed. São Paulo: McGraw-Hill, 2006.
- Project Management Institute – PMI. Um Guia do Conhecimento em Gerenciamento de Projetos – Guia PMBOK. 4. ed. Pennsylvania, EUA, 2008.
- ROYCE, W.W. *Managing the development of large software systems*. Proc. IEEE WESCON, Aug. 1970.
- SCHWABER, K. *Agile Project Management with Scrum*. Washington: Microsoft Press, 2004.
- SCHWABER, K. *Scrum But*. ScrumAlliance, 2008.
- SLIGER, M.; BRODERICK, S. *The Software Project Manager's Bridge to Agility*. Boston: Addison-Wesley Professional, 2008.
- TAKEUCHI, H.; NONAKA, I. *The New New Product Development Game*. Harvard Business Review, p. 137-146, jan-fev 1986.