

# Desenvolvimento Ágil de Software

- Mentalidade Ágil
- Processos ágeis
- SCRUM

# Objetivos

- Apresentar a importância da mentalidade ágil no contexto contemporâneo
- Apresentar e discutir os modelos de desenvolvimento ágil

# Referências Bibliográficas

- Básica:

PRESSMAN, Roger S. Engenharia de Software: uma abordagem profissional. 8 ed. AMGH, 2016.

- Complementar:

SOMMERVILLE, Ian. Engenharia de software. 10 ed. São Paulo: Pearson, 2019.

# O que é agilidade?

- Eficácia (rápido e adaptativo) para resposta a mudanças
- Eficácia na comunicação entre os *stakeholders* (envolvidos)
- Envolvimento do cliente na equipe de desenvolvimento
- Organização da equipe para entregar incrementos de software

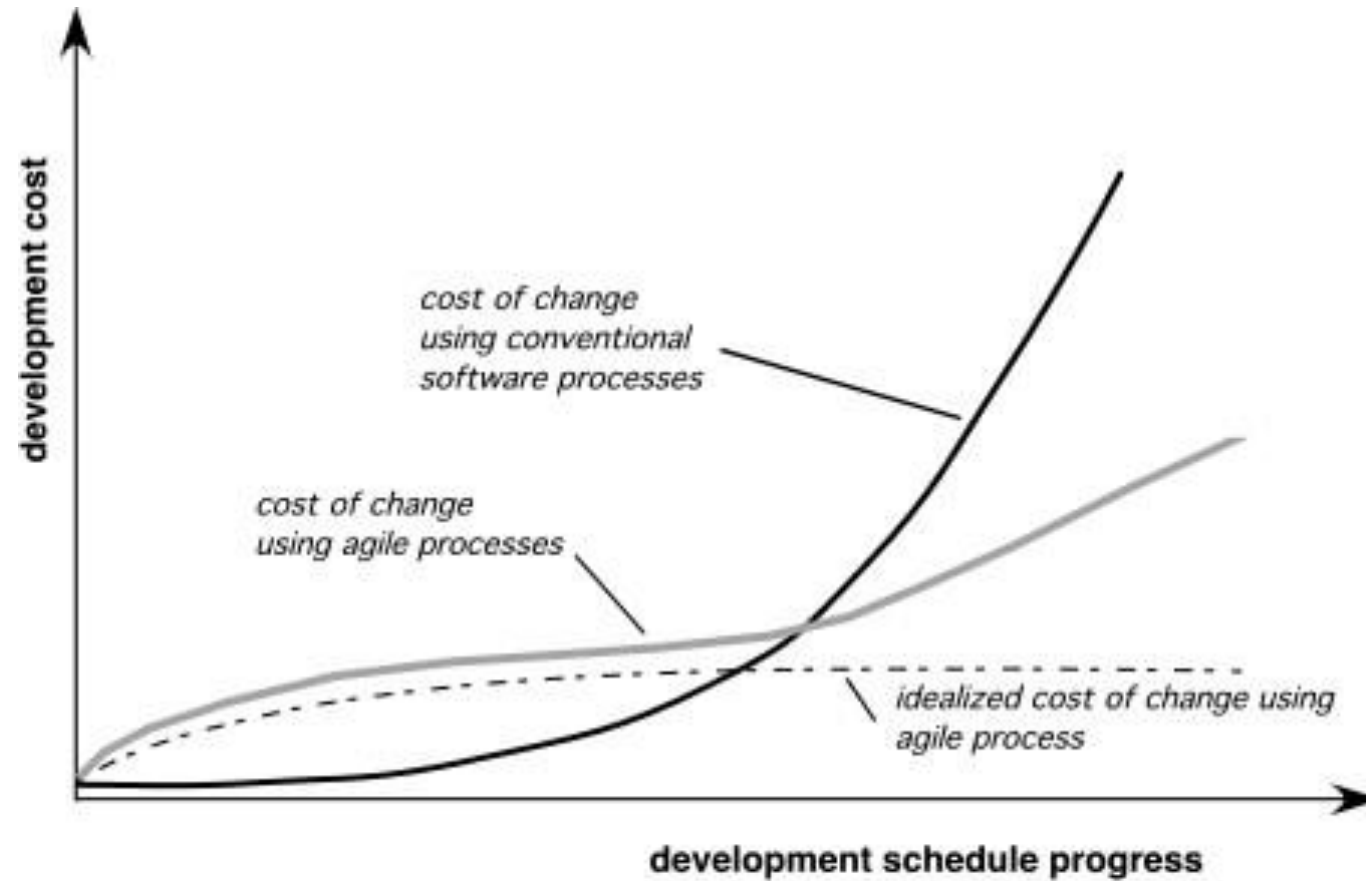
*Produzindo...*

- rapidamente, software de forma incremental

# Agilidade para...

*... produzir e entregar rapidamente incrementos de software!*

# Agilidade e o custo da mudança



(PRESSMAN, 2016. p. 69)

# O Manifesto do Desenvolvimento de Software Ágil (Kent Beck et. al., 2001)

“Desenvolvendo e ajudando outros a desenvolver software, estamos desvendando formas melhores de desenvolvimento. Por meio deste trabalho, passamos a valorizar:

- Indivíduos e interação acima de processos e ferramentas
- Software em funcionamento acima de documentação abrangente
- Colaboração com o cliente acima de negociação de contratos
- Responder a mudanças acima de seguir um plano

Ou seja, embora haja valor nos itens à direita, valorizaremos os da esquerda mais ainda.”

# Em síntese...

- Faça da comunicação um pré-requisito
- Entregue valor a todo instante
- Torne as pessoas sensacionais
- Experimente e aprenda rápido



# Princípios da agilidade da *Agile Alliance*

1. A maior prioridade é satisfazer o cliente por meio de entrega adiantada e contínua de software valioso
2. Acolha bem os pedidos de alterações, mesmo atrasados no desenvolvimento; os processos ágeis se aproveitam das mudanças como uma vantagem competitiva na relação com o cliente
3. Entregue software em funcionamento frequentemente, de algumas semanas para alguns meses, dando preferências a intervalos mais curtos

# Princípios da agilidade da *Agile Alliance*

4. O pessoal comercial e os desenvolvedores devem trabalhar em conjunto diariamente ao longo de todo o projeto
5. Construa projetos em torno de indivíduos motivados; dê a eles o ambiente e apoio necessários e confie neles para ter o trabalho feito
6. O método mais eficiente e efetivo de transmitir informações para e dentro de uma equipe de desenvolvimento é uma conversa aberta, de forma presencial

# Princípios da agilidade da *Agile Alliance*

- 7. Software em funcionamento é a principal medida de progresso
- 8. Os processos ágeis promovem desenvolvimento sustentável; os proponentes, desenvolvedores e usuários devem estar capacitados para manter um ritmo constante indefinidamente
- 9. Atenção contínua para com a excelência técnica e para com bons projetos aumenta a agilidade

# Princípios da agilidade da *Agile Alliance*

- 10. Simplicidade – a arte de maximizar o volume de trabalho não efetuado – é essencial
- 11. As melhores arquiteturas, requisitos e projetos emergem de equipes que se auto organizam
- 12. A intervalos regulares, a equipe se avalia para ver como tornar-se mais eficiente, então sintoniza e ajusta seu comportamento de acordo com o que seja necessário

# Política do Desenvolvimento Ágil

- Grande discussão acerca das metodologias tradicionais versus metodologias ágeis
- Não se pode apontar melhor ou pior
- O ideal é aplicar o melhor de cada uma para cada contexto

# Fatores Humanos (1/2)

- O processo se adapta às necessidades das pessoas e equipes específicas, e não o caminho inverso
- Deve haver traços chave entre as pessoas de uma equipe ágil e a equipe em si:

(...)

# Fatores Humanos (2/2)

- Competência
- Foco comum
- Colaboração
- Habilidade na tomada de decisão
- Habilidade de solução de problemas confusos
- Confiança mútua e respeito
- Auto-organização

# Processos Ágeis



# Processo Ágil

- Dirigido por cenários construídos pelos clientes com seus requisitos
- Planejamento tem vida curta
- Desenvolvimento iterativo com uma forte ênfase nas atividades de construção
- Entrega de múltiplos incrementos de software
- Adaptação às mudanças

# *Extreme Programming (XP)*

- Também conhecido como Programação Extrema
- Foi proposto por Kent Beck em 2004
- Processo ágil mais utilizado

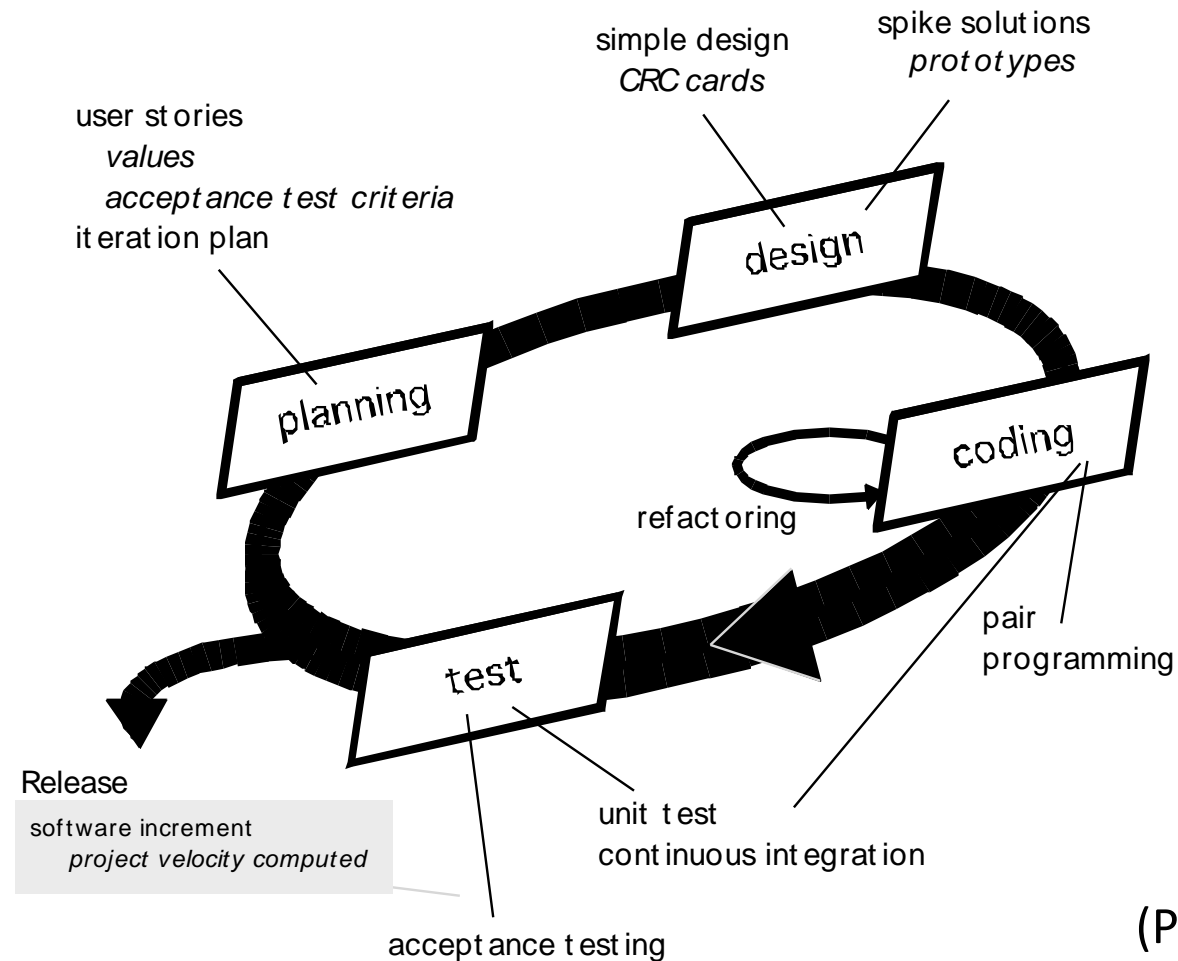
# Valores básicos do XP (1/2)

- Comunicação
  - Relação estreita e informal
- Simplicidade
  - Fazer somente o necessário
- Feedback
  - Devem ser constantes, à medida do desenvolvimento, com testes e entregas

# Valores básicos do XP (2/2)

- Coragem (ou disciplina)
  - Disciplina para pensar somente no presente, mas bem feito
- Respeito
  - Conforme as entregas vão acontecendo, o respeito pela equipe e pelo software vai aumentando

# Extreme Programming



(PRESSMAN, 2016. p. 69)

# Processo XP: *Planejamento*

- Comece com a criação das “histórias dos usuários”
- A equipe avalia as histórias e estima o custo
- As histórias são agrupadas em um incremento de software
- Define-se uma data de entrega do incremento
- O primeiro incremento é utilizado como referência para acelerar os demais

# Processo XP: *Desenho*

- Segue o princípio KISS (*keep it simple*)
- Encoraja o uso de cartões CRC
- Situações de dificuldade são resolvidas com prototipação
- Encoraja a *refatoração* – refinamento iterativo do desenho interno do programa

# Processo XP: *Codificação*

- Recomenda a construção de testes de unidade antes do início da codificação
- Encoraja a programação em par



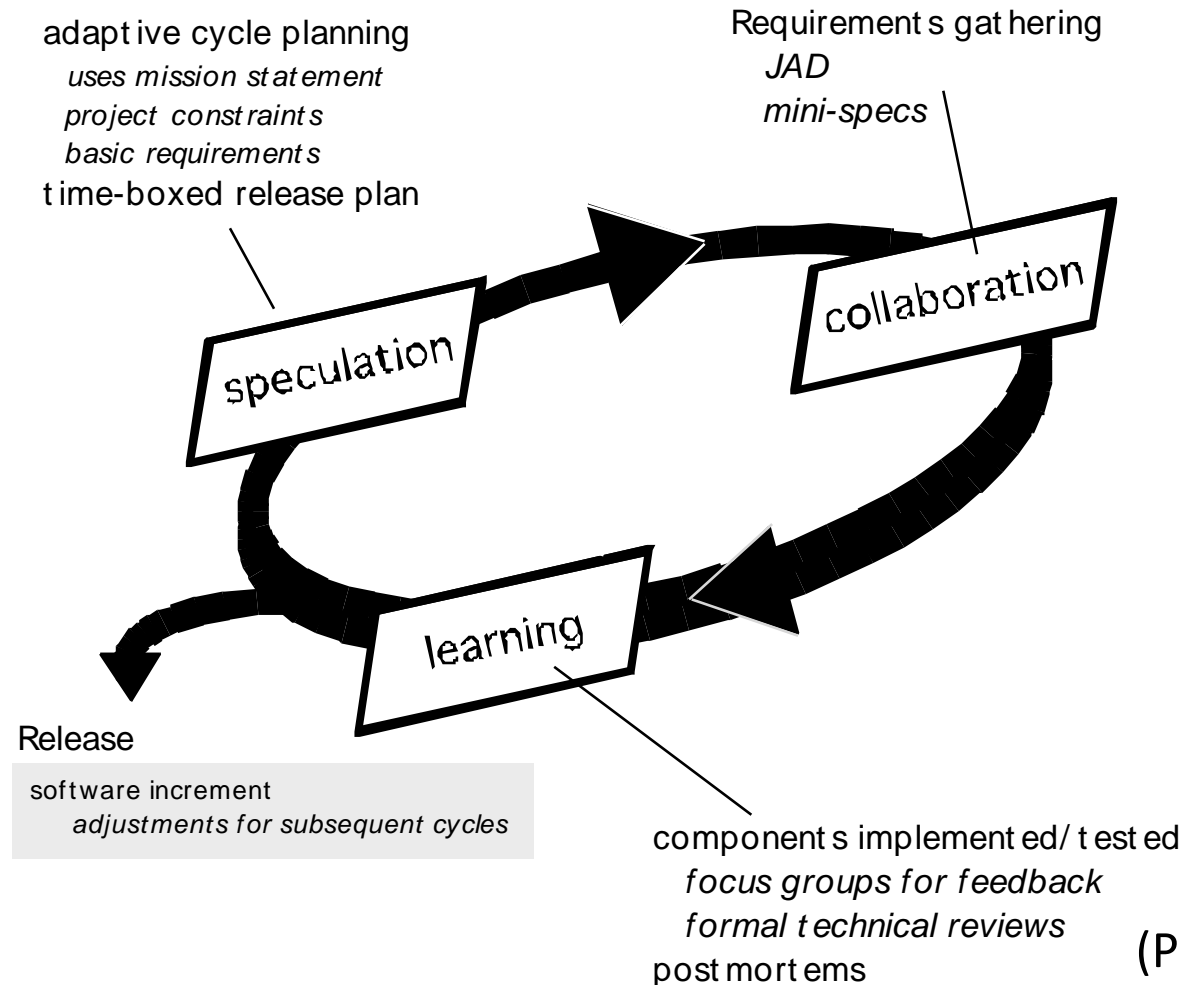
# Processo XP: *Teste*

- Todos os testes de unidade são executados diariamente
- Testes de aceitação são definidos pelo cliente para avaliação de uma funcionalidade

# Industrial XP (IXP)

- Evolução orgânica do XP
  - Avaliação imediata
  - Comunidade de projeto
  - Mapeamento de projeto
  - Gerenciamento orientado a testes
  - Retrospectivas
  - Aprendizagem contínua

# Desenvolvimento de Software Adaptativo



(PRESSMAN, 2016. p. 69)

# Desenvolvimento de Software Adaptativo

- Proposto por Jim Highsmith em 2000
- Características do ASD (*Adaptive Software Development*)
  - Planejamento dirigido a missão
  - Foco no desenvolvimento baseado em componentes
  - Uso de “time-boxing”
  - Consideração explícita de riscos
  - Ênfase na colaboração para entendimento dos requisitos
  - Ênfase na “aprendizagem” durante o processo

# Método Dinâmico de Desenvolvimento de Sistemas (1/3)

- Promovido pelo consórcio DSDM – *Dynamic Systems Development Method*, [www.dsdm.org](http://www.dsdm.org))
- Características do DSDM
  - Similar ao XP e ao ASD em muitos aspectos

# Método Dinâmico de Desenvolvimento de Sistemas (2/3)

- Princípios:

1. O envolvimento ativo do usuário é imperativo
2. As equipes devem ter autonomia para tomada de decisão
3. O foco é a entrega frequente de produtos
4. Conveniência para o negócio é um critério essencial para aceitação da entrega

# Método Dinâmico de Desenvolvimento de Sistemas (3/3)

- Princípios:

5. Desenvolvimento iterativo e incremental é necessário para convergência de uma solução de negócio correta
6. Todas as mudanças durante o desenvolvimento são reversíveis
7. Requisitos são marcos de alto nível
8. Testes são integrados ao longo do ciclo de vida

# Crystal

- Proposto por Cockburn e Highsmith em 2002
- Características
  - Permite a adaptabilidade baseada nas características do problema
  - Ênfase na comunicação face a face
  - Sugere-se o uso de workshops de reflexão para revisão da forma de trabalho da equipe



# Desenvolvimento Direcionado por Características

- Proposto por Peter Coad em 1999
- Características do FDD (*Feature Driven Development*)
  - Ênfase na definição de características (*feature*), que é “uma função valorizada pelo cliente que pode ser implementada em duas semanas ou menos”

# Desenvolvimento Direcionado por Características

- Mais características do FDD
  - Utiliza o seguinte *template*  
*<action> the <result> <by | for | of | to> a(n) <object>*
  - A lista de características é criada e o planejamento por características é conduzido
  - O desenho se funde com a construção no FDD

# Processo Unificado Ágil (AUP)

- Adota a filosofia:
  - Serial para o que é amplo
  - Iterativo para o que é particular

# Atividades do AUP

- Modelagem
- Implementação
- Teste
- Aplicação (entrega)
- Configuração e gerenciamento de projeto
- Gerenciamento do ambiente

# Scrum

# Declaração de Interdependência (DOI) da Gestão de Projeto Ágil (2005) (1/2)

“Somos uma comunidade de líderes de projeto que tem sido altamente bem-sucedida em entregar resultados. Para alcançar tais resultados:”

- ***Aumentamos o retorno de investimento***, tornando o fluxo contínuo de valor o nosso foco
- ***Entregamos resultados confiáveis***, engajando interações frequentes e propriedade compartilhada
- ***Esperamos incertezas*** e gerenciamos levando-as em conta, por meio de interações, antecipação e adaptação

# Declaração de Interdependência (DOI) da Gestão de Projeto Ágil (2005) (2/2)

- ***Promovemos criatividade e inovação reconhecendo*** que os indivíduos são a fonte última e criamos um ambiente em que eles fazem a diferença
- ***Impulsionamos o desempenho*** por meio do compromisso do grupo em obter resultados e da responsabilidade compartilhada pela eficácia do grupo
- ***Melhoramos a eficácia e a confiabilidade*** por meio de estratégias situacionais específicas, processos e práticas.

# Origem do Scrum

TAKEUCHI, Hirotaka, NONAKA, Ikujiro. The new new product development game. *Harvard Business Review*, January 1986 Issue.

“... equipes de projeto são compostas de pequenas equipes multifuncionais, trabalhando com sucesso rumo a um objetivo comum, que os autores compararam à formação Scrum do *rugby*.”



# Origem do Scrum



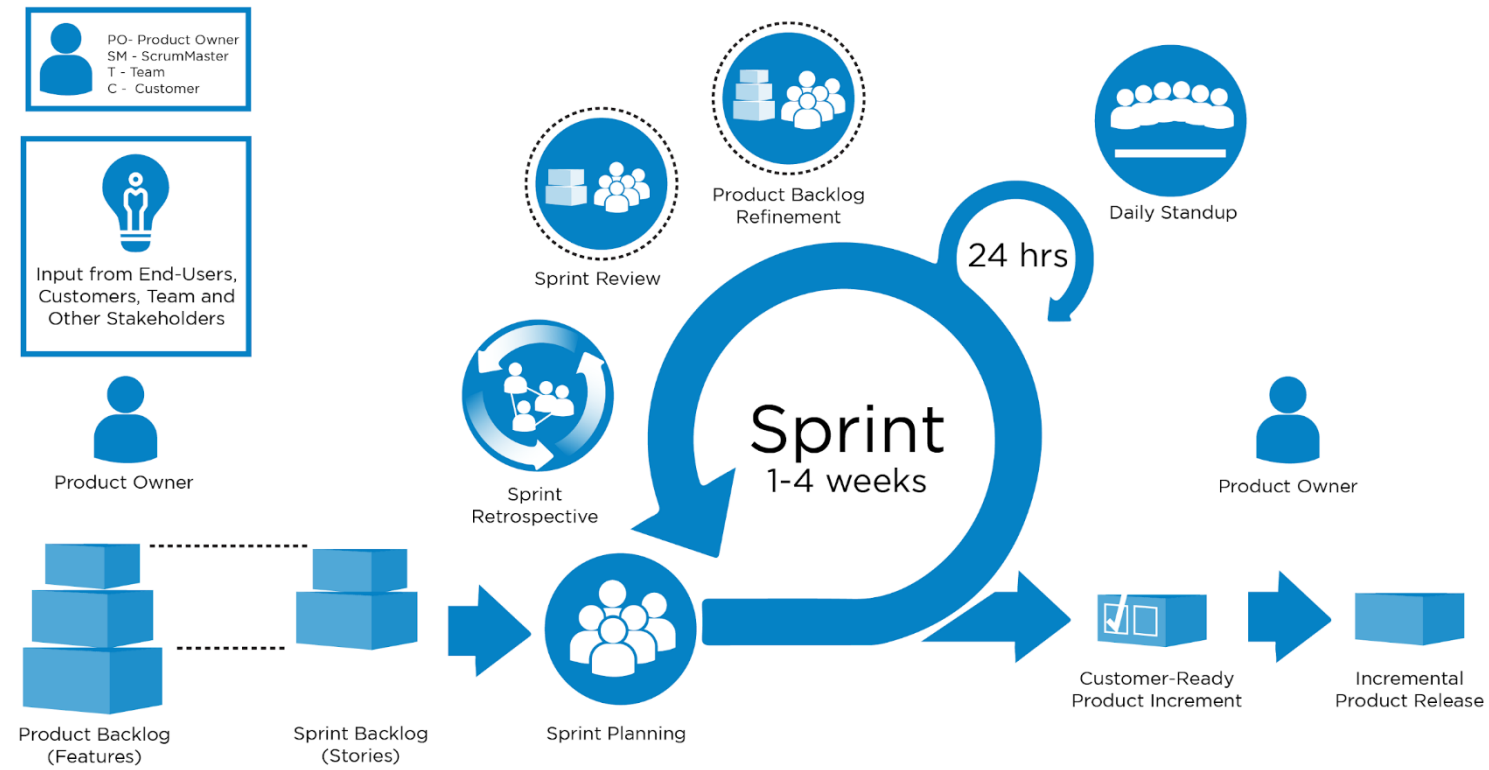
# Origem do Scrum

- A pedido do *Object Management Group* (OMG), **Jeff Sutherland** e **Ken Schwaber**, em 1995, publicaram:

*Scrum and the perfect storm*  
(Scrum e a tempestade perfeita)

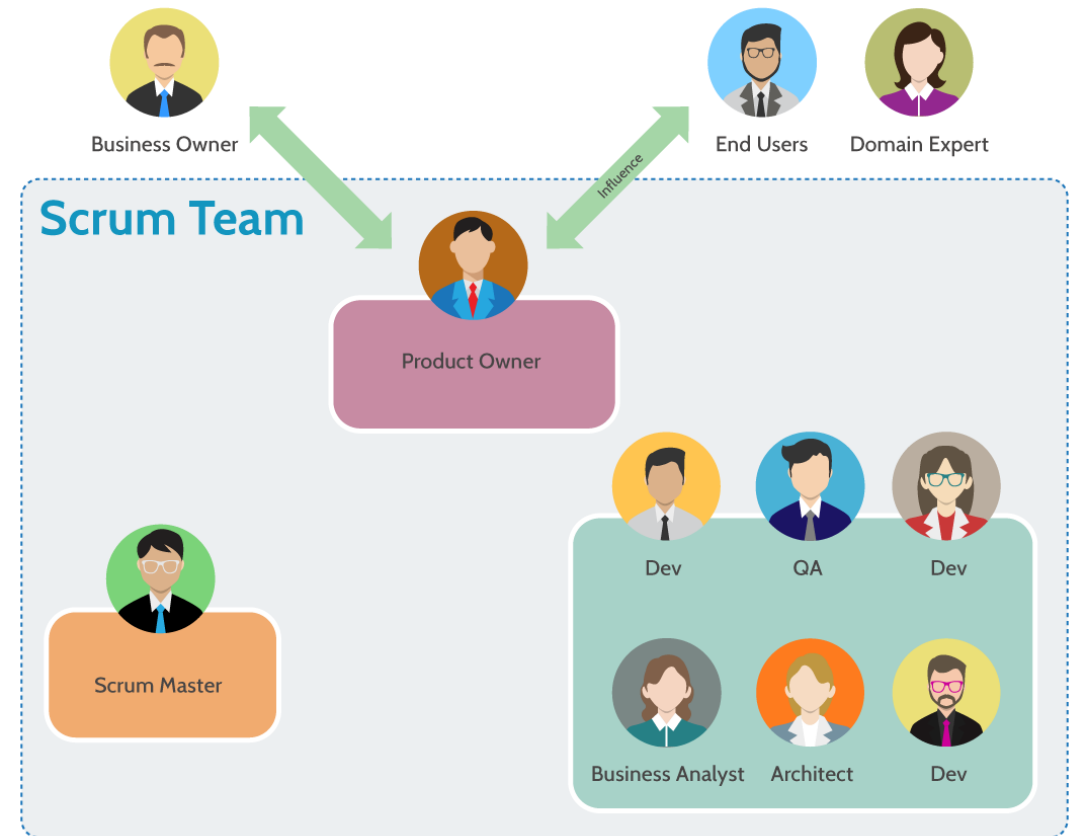
# Como funciona?

## How Scrum Works

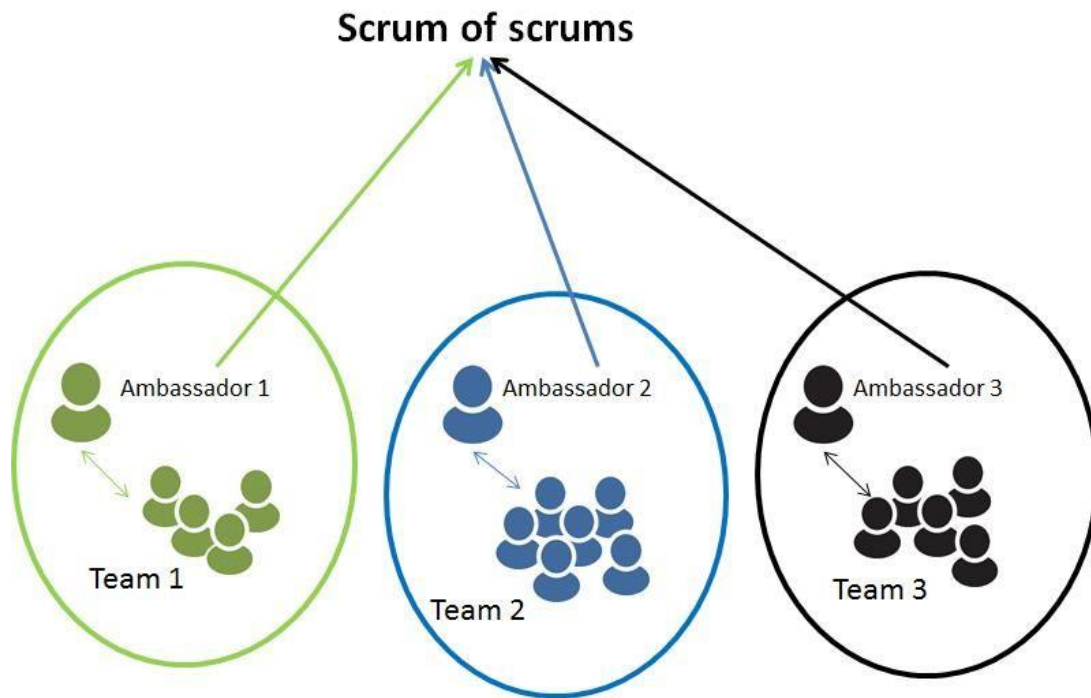


# Papéis fundamentais do Scrum

- *Product Owner (PO)*
- *Scrum Master (SM)*
- *Scrum Team (ST)*



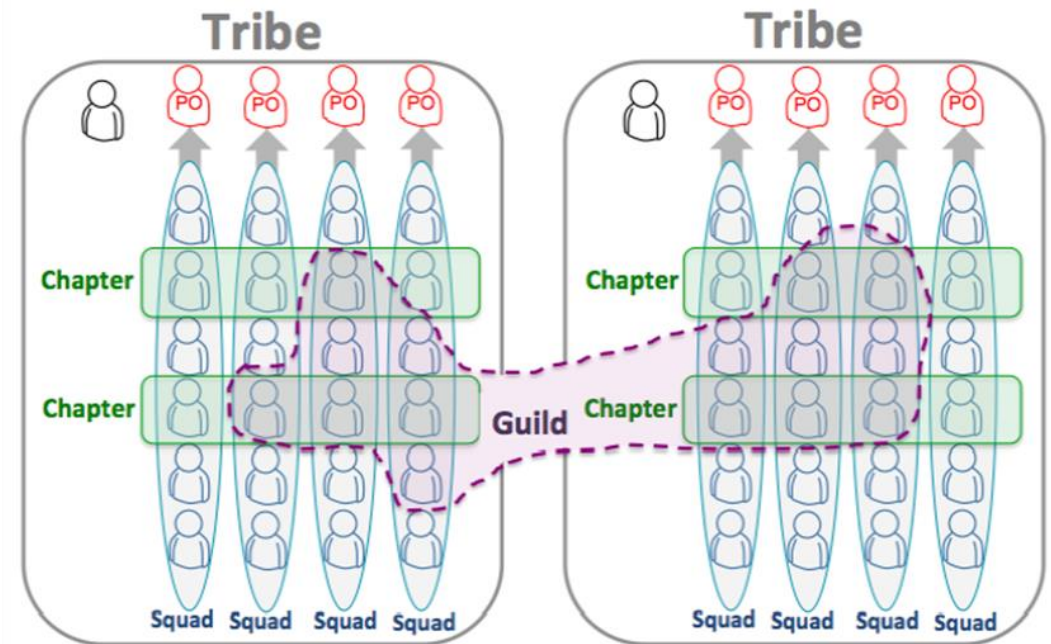
# E vai crescendo...



## Scaling Agile @ Spotify

with Tribes, Squads, Chapters & Guilds

Henrik Kniberg & Anders Ivarsson  
Oct 2012



# Início de um Projeto Scrum

- Um projeto Scrum é iniciado quando o *Product Owner* (PO), responsável por obter informações dos *stakeholders*, ou usuários que os representem, elaboram uma lista de requisitos e criam um *Backlog* de Produto.





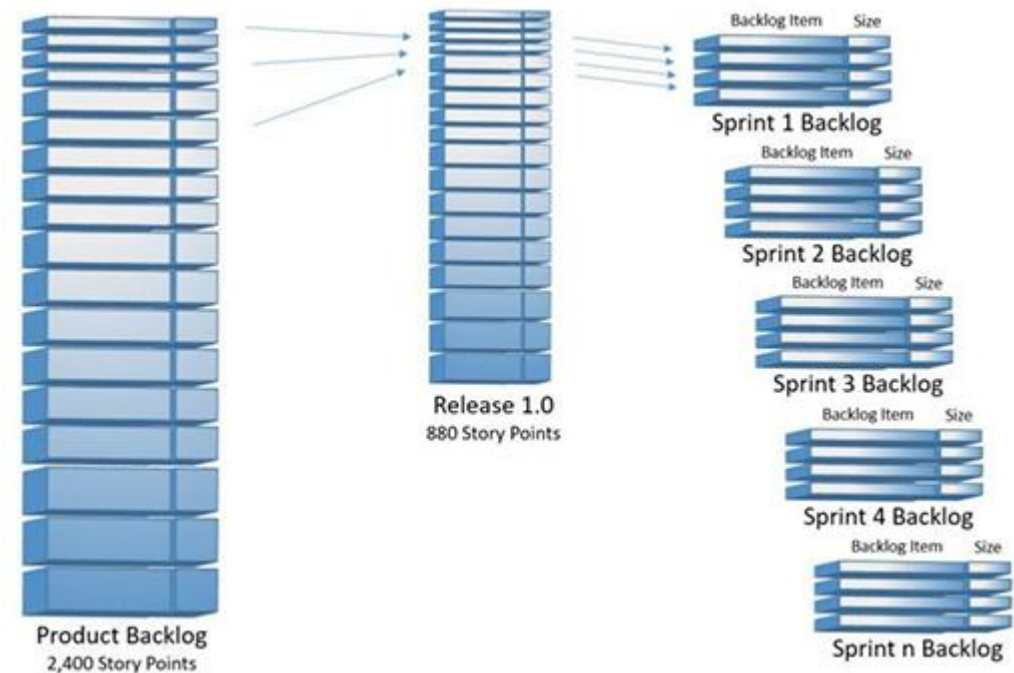
# O que é um *Backlog* de Produto?

- Lista de requisitos priorizada, que pode incluir de tudo:
  - aspectos do negócio a tecnologias
  - questões técnicas
  - correções de bugs



# Criação de um *Backlog* de Produto

- Os requisitos dos usuários costumam ser coletados como histórias de usuários curtas
- Workshop de um ou dois dias, anterior à reunião de Planejamento de Releases e à reunião de Planejamento de Sprints





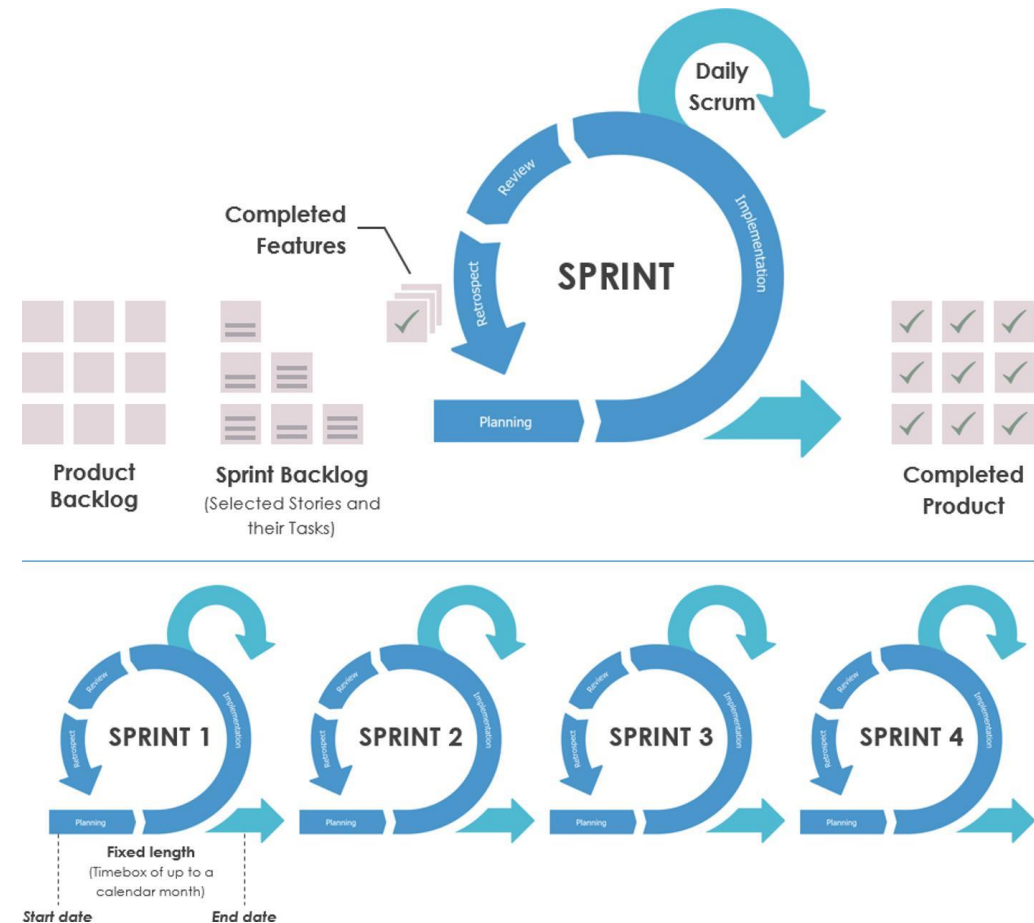
# Planejamento de Releases

- Socialização da agenda de entregas prováveis
- Duração: cerca de quatro horas para cada Sprint de duas ou quatro semanas



# Planejamento de Sprints

- Duração: cerca de oito horas para cada Sprint de quatro semanas (ou quatro horas para Sprint de duas semanas)
- Geralmente é dividido em duas reuniões de quatro horas



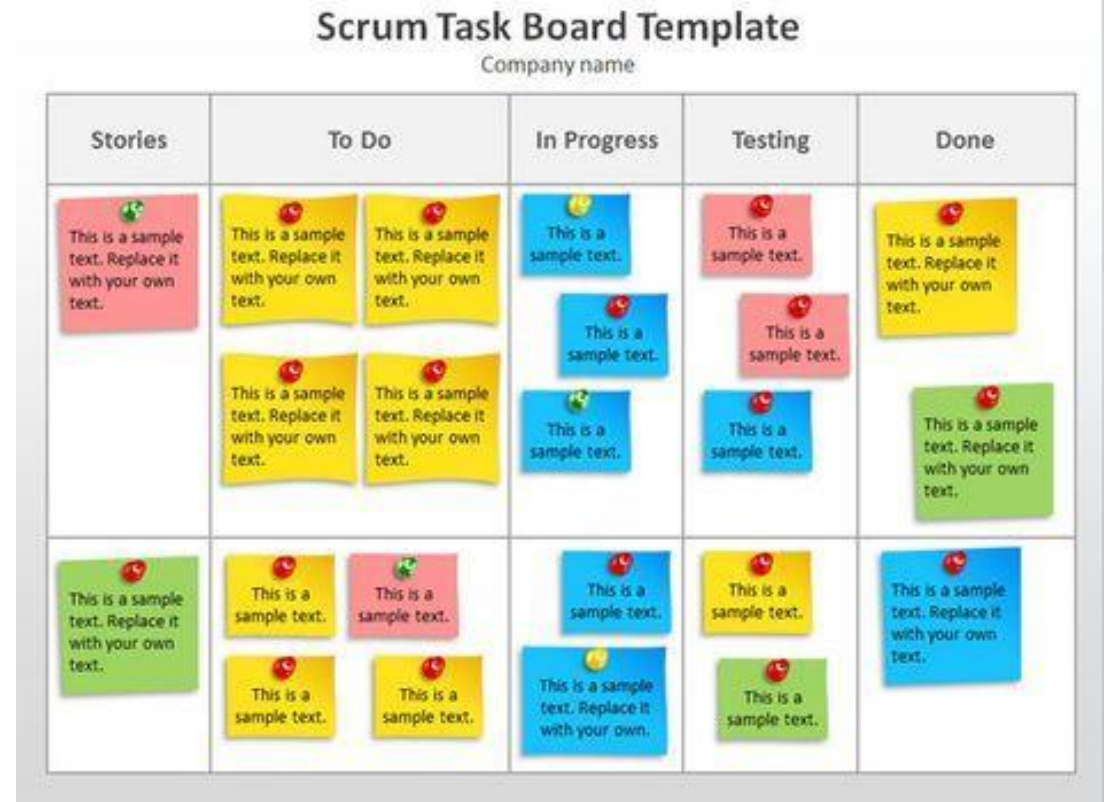
# O que dever ser feito na Sprint?

- O *Product Owner* organiza, com a participação colaborativa da equipe, os requisitos (histórias dos usuários) e definem, também juntos, os objetivos
- Cria-se um *Backlog* de Sprint, que, geralmente, não pode ser modificado

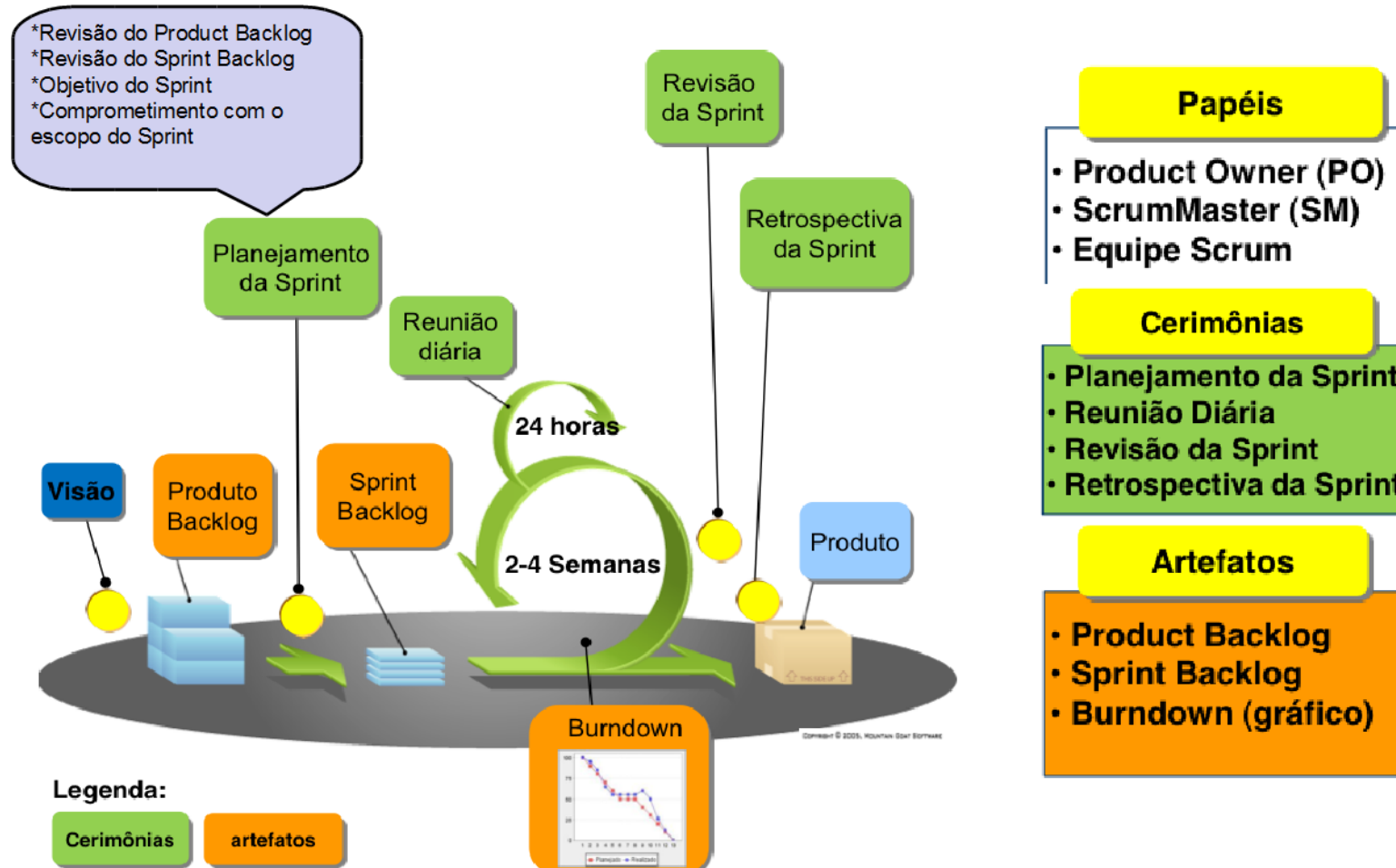


# Como a Sprint deve ser realizada?

- Identificação de tarefas com base nas histórias dos usuários
- Estimativa de tempo para os incrementos que serão entregues
- Construção de um Quadro de Tarefas (*Task Board*)

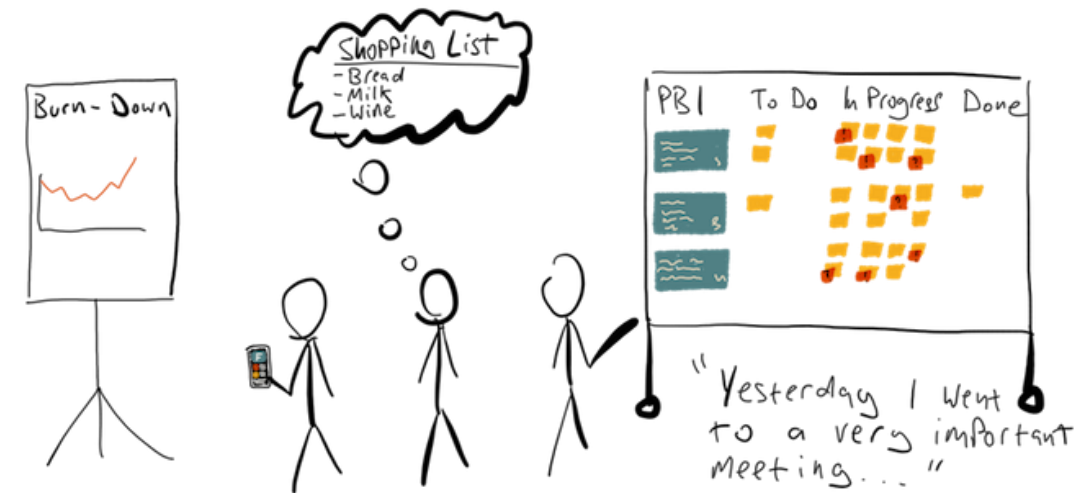


# Visão geral da Sprint



# Reuniões Diárias (*Daily Meetings*)

- *Daily Scrum* ou *Daily Standup Meeting*
- Duração: 15 minutos
- Pauta:
  - O que foi feito desde a última reunião?
  - Quais foram os obstáculos?
  - O que será feito?
- Atualização do *Gráfico de Burndown*





# *Remote Daily Meeting?*



# Reunião para Revisão da Sprint

- Duração: cerca de uma hora para cada semana de Sprint
- Realizada com o PO para:
  - a) Relatar o que foi feito para obter feedback
  - b) Apresentar e explicar o que não foi feito
  - c) Atualizar o *roadmap* do produto





# Reunião para Retrospectiva da Sprint

- Duração: cerca de uma hora para cada semana de Sprint
- O que funcionou e o que não funcionou durante o Sprint atual?
- O que pode melhorar no próximo Sprint?



# Responsabilidades

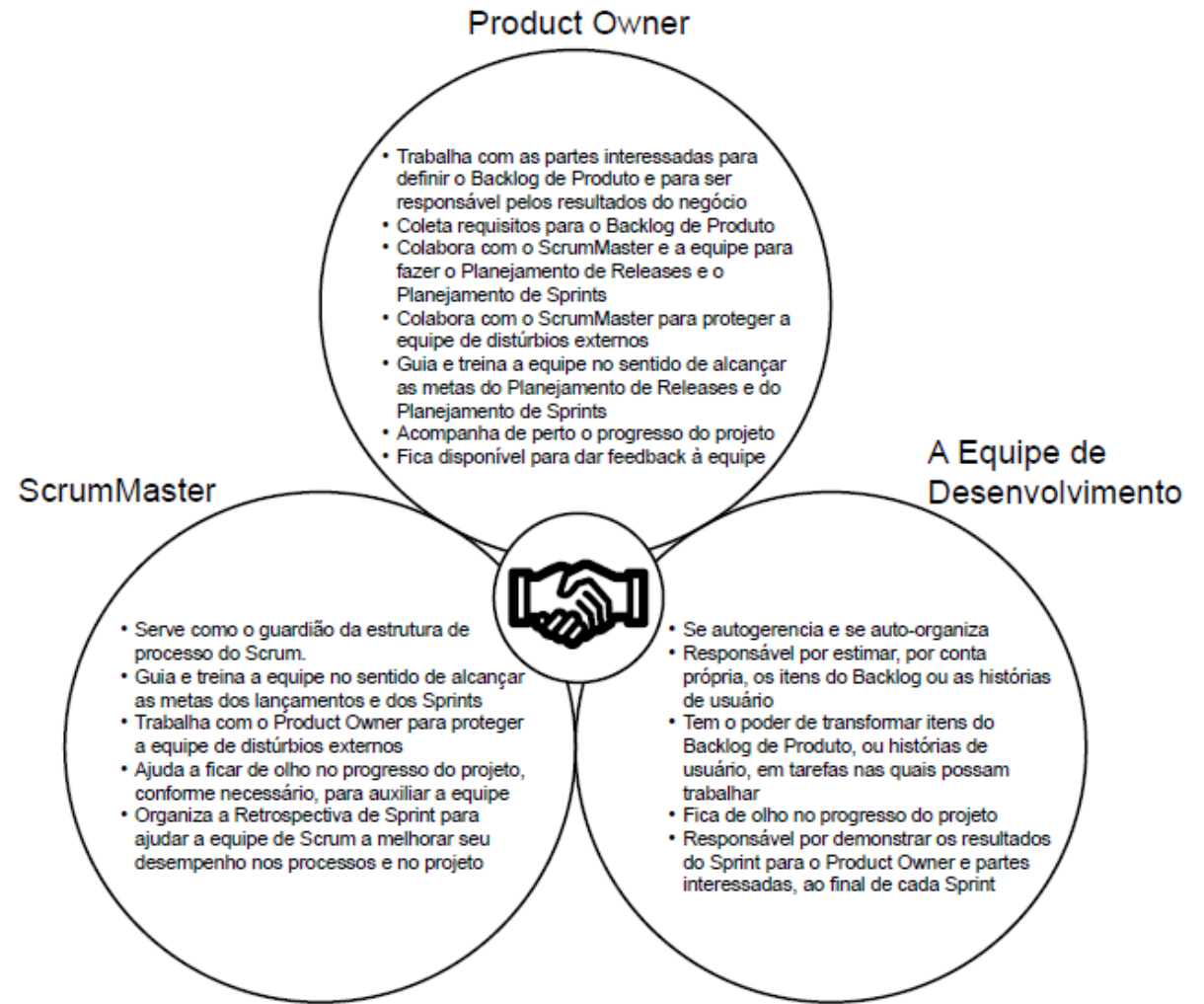
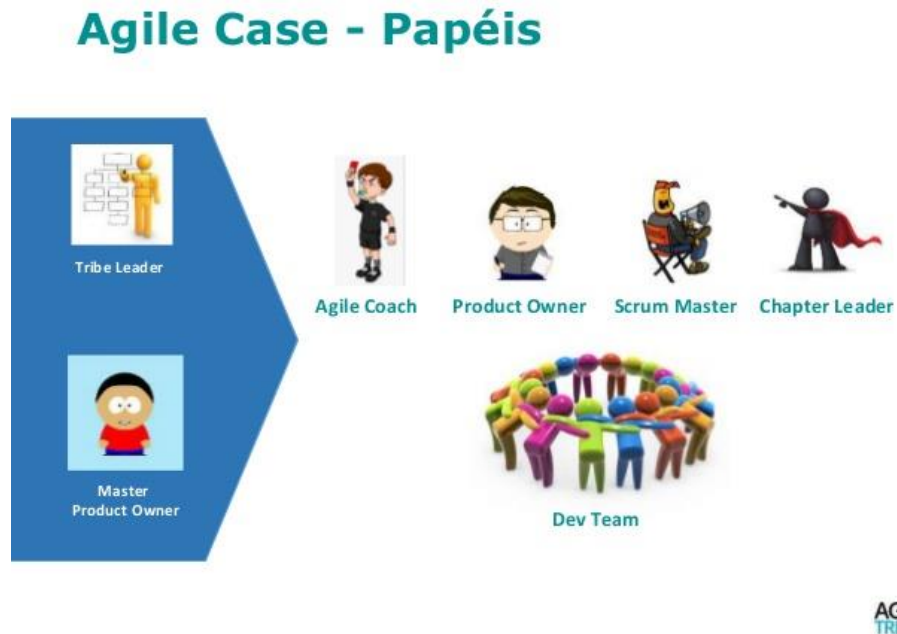
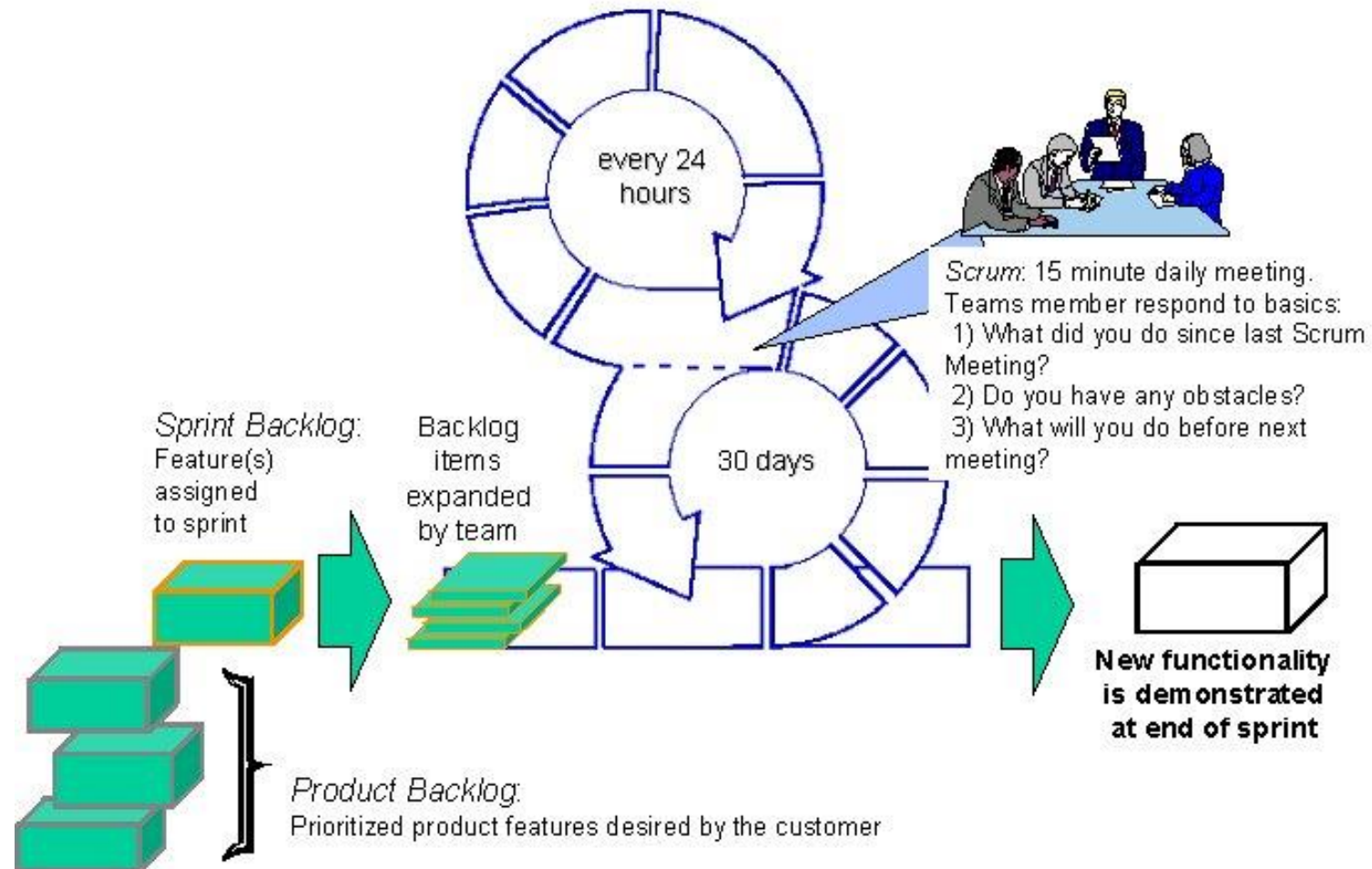


Figura 14 – Tudo se resume à colaboração entre a equipe, o ScrumMaster e o Product Owner.

# Scrum Process Flow



# Scrum Process Flow

## The Agile: Scrum Framework at a glance

Inputs from Executives,  
Team, Stakeholders,  
Customers, Users



Product Owner



The Team



Product Backlog



Sprint Planning Meeting



Sprint Backlog



# Kanban

# Origem do Kanban

- Palavra de origem japonesa que significa registro ou placa visível
- Sistema de escalonamento para produção “enxuta” e “*just-in-time*”.
- Sistema utilizado para controle de cadeia logística ao invés de controle de estoque
- Desenvolvido pela Toyota em 1953

# Como funciona?

- Os cartões (e.g. *post-it*) são utilizados para indicar o fluxo de uma produção em série
- Cada tarefa está associada a um cartão



# Como funciona?

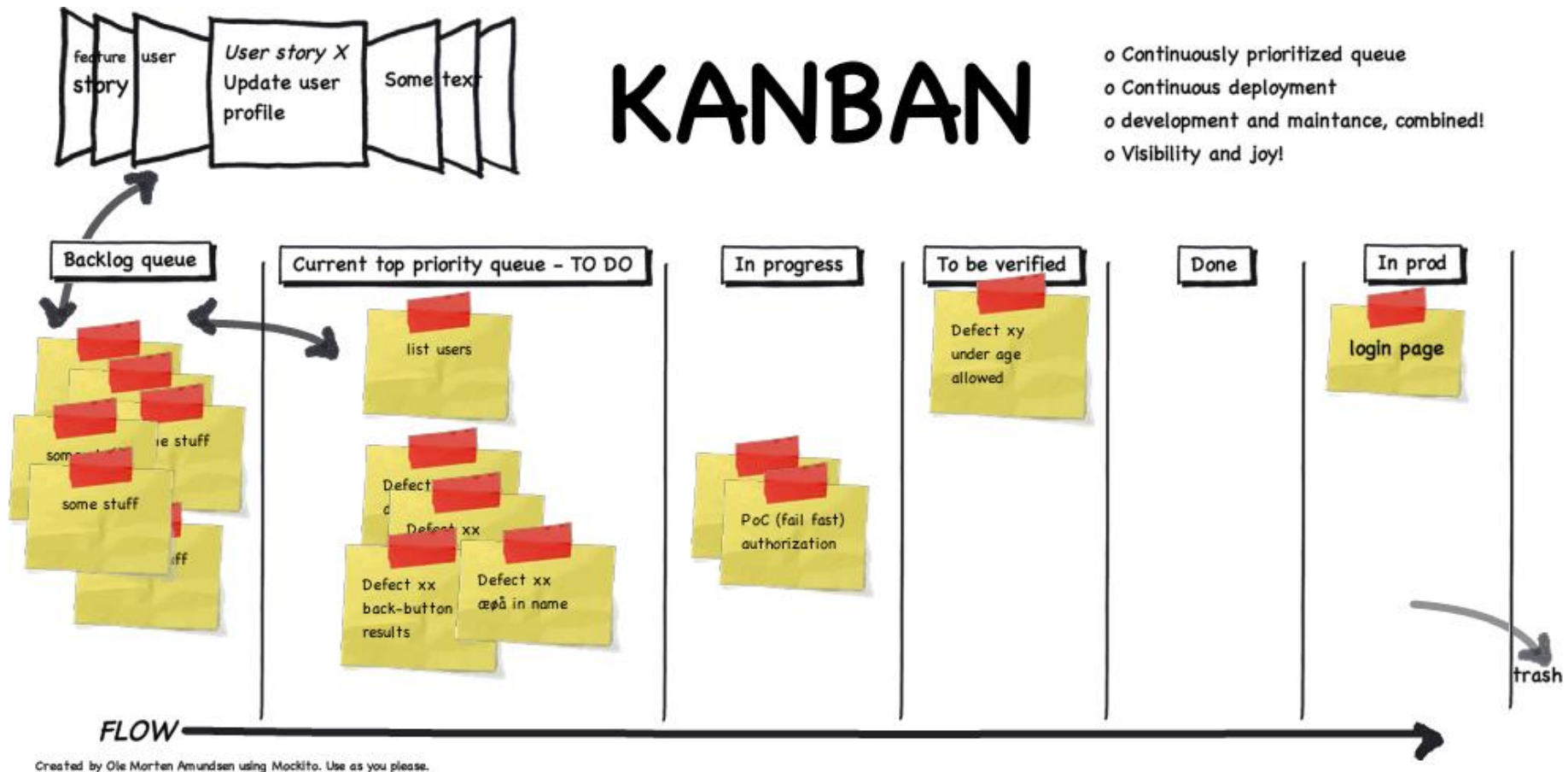
- Se resume em três etapas:
  1. Visualizar processos
  2. Limitar o trabalho em processo (*WIP – Work In Progress*)
  3. Gerenciamento do *lead-time*, ou seja, tempo que a atividade leva para passar por todas as fases até a sua entrega






# Implementação do Modelo Kanban

- O número de atividades ou cartões em circulação é equivalente à capacidade de sistema
- Conceito de “puxar tarefa” quando há capacidade de processá-la
- Em software: *requisitos são adicionados à lista de backlog e “puxados” por um membro da equipe*

# Um exemplo



# Outro exemplo

backlog	Análise 2	Dev 3	Teste 3	Aprovação 1	Finalizado
<div>Fazer o item de recolhimento...</div> <div>Fazer o item de recolhimento...</div> <div>Fazer o item de recolhimento...</div> <div>Fazer o item de recolhimento...</div>	<div>Fazer o item de recolhimento...</div> <div>Fazer o item de recolhimento...</div>	<div>Fazer o item de recolhimento...</div> <div>Fazer o item de recolhimento...</div> <div>Fazer o item de recolhimento...</div>	<div>Fazer o item de recolhimento...</div> <div>Fazer o item de recolhimento...</div> <div>Fazer o item de recolhimento...</div>	<div>Fazer o item de recolhimento...</div>	<div>Fazer o item de recolhimento...</div>
	Flávio				
	Marina				
	Leandro				

# Mais um exemplo

MODUS COOPERANDI BOARD 2013						
Active Work In Progress						
Priority 1	Priority Toni	Today	Doing		The Pen	Done
<p>Blog post about release groups - definitions around functional releases not time based</p> <p>Make this a blog post Each team has a personality, a penchant, a patron, a purchaser,</p> <p>Pomodoro A3 Personas</p>	<p>Respond to <del>...</del></p> <p>Finish Element 15</p> <p>Discuss / Return Kidzban information</p>	<p>Pomodoro Why Limit WIP?</p>	<p>Jim Doing</p> <p>Send out Seattle List</p> <p>Write Pomodoro for Productive Focus Post</p>	<p>Toni Doing</p> <p>Procrastination Research</p>	<p>1099s</p> <p>Ask <del>...</del> about doing a class like you discussed with Ryan and . last year</p> <p>Contact <del>...</del></p> <p>Contact <del>...</del></p> <p>Contact <del>...</del></p>	<p>City of Seattle License</p> <p>Do a <del>...</del> in <del>...</del></p> <p>Add Contacts from <del>...</del></p> <p>Check out <del>...</del>'s contact and ... contact</p> <p>Send review copies to positively responding reviewers</p>

# Por que usar?

- Manter, medir, otimizar e visualizar um fluxo constante de trabalho
- Limitar a quantidade de trabalho em andamento e reduzir estresse da equipe
- Melhorar a previsibilidade e a colaboração
- Contribuir para a maturidade da equipe

# Quando *não* usar métodos ou processos ágeis?

*Fonte:* Embracing Agile. Rigby, Sutherland & Takeuchi. Harvard Business Review, 2016

- Condições do mercado são estáveis e previsíveis
- Requisitos são claros no início do projeto e irão permanecer estáveis
- Clientes não estão disponíveis para colaboração frequente
- Sistema semelhante já foi feito antes; logo, já se conhece a solução
- Problemas podem ser resolvidos sequencialmente, em silos funcionais
- Clientes não conseguem testar partes do produto, antes de completo
- Mudanças no final do projeto são caras ou impossíveis
- Impacto de mudanças provisórias pode ser catastrófico