DCC / ICEx / UFMG

Desenvolvimento Ágil de Software

Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo

Agenda

- Métodos ágeis
  - Histórico e Motivação
  - Manifesto ágil
- Desenvolvimento dirigido a planos e ágil
- Programação Extrema
- Scrum

Métodos ágeis

Histórico e Motivação

- 1960s/1970s: crise do software
  - Problemas com orçamento
  - Problemas com prazos
  - Problemas com qualidade
  - Problemas com requisitos
  - Problemas com manutenibilidade
  - Outros problemas...

1980s/1990s: ES tradicional

- Planejamento cuidadoso
- Qualidade formalizada
- Uso de métodos de análise e projeto
- Ferramentas CASE (*Computer-Aided Software Engineering*)
- Processo de desenvolvimento rigoroso e controlado

1980s/1990s: ES tradicional ...

- Desenvolvimento de software grandes e críticos (ex: sistemas aeroespaciais e de governo)
- Sistemas duradouros, precisam ter boa manutenibilidade
- Grandes/diferentes empresas
- Grandes equipes, podendo estar dispersas geograficamente
- Longos períodos (até 10 anos)

2000s/2010s: ES ágil

- Novas oportunidades, novos mercados, novos produtos, novos concorrentes
- As regra de negócios mudam rapidamente
  - O software tem que ser adaptado para as novas regras
- Importância do software nesse cenário: precisam ser desenvolvidos rapidamente, com agilidade
  - A entrega rápida pode ser tão (ou mais) desejável que a qualidade
- Processos “pesados” causam muito *overhead*

Métodos ágeis

- Ainda no final da década de 1990, desenvolvedores propuseram processos mais “leves”: os “métodos ágeis”
- Métodos ágeis tem por objetivo criar software util rapidamente
  - Não se preocupam com a documentação completa em todas as fases
  - O sistema de software é desenvolvido por uma série de incrementos, cada incremento, uma nova funcionalidade

Manifesto ágil, 2001

"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."

Princípios dos métodos ágeis

- Envolvimento do cliente
- Entrega incremental
- Pessoas, não processos
- Aceitar as mudanças
- Manter a simplicidade

Dificuldades/limitações

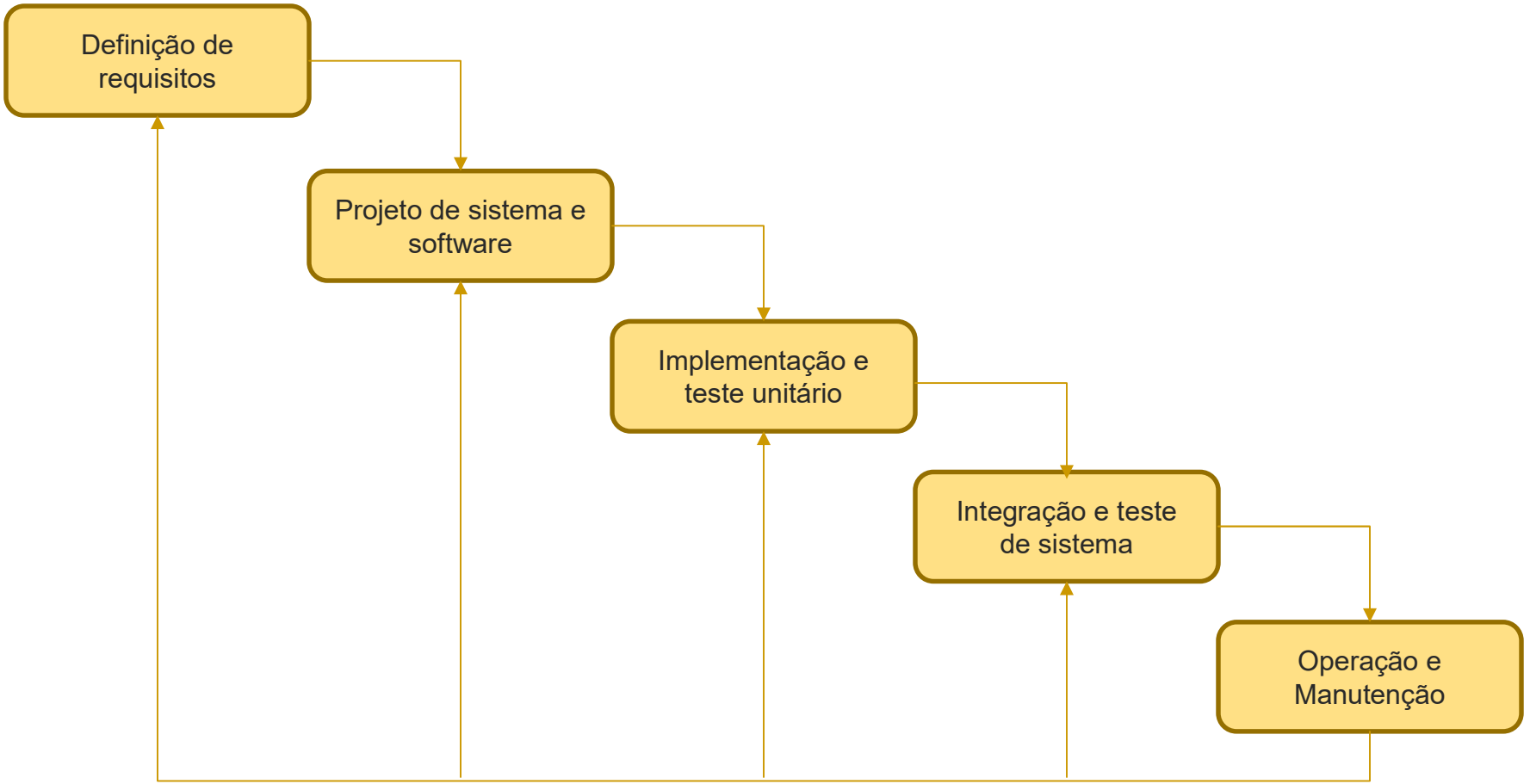
- Clientes deve estar disposto e capaz de passar o tempo com a equipe de desenvolvimento.
- Membros individuais da equipe podem não interagir bem com outros membros da equipe.
- Priorizar as mudanças pode ser extremamente difícil.
- Manter a simplicidade pode ser muito complexo.
- A organização pode não ter a cultura adequada.
  - Organização investiram muito em definição e organização de processos.

Desenvolvimento dirigido a planos e ágil

Desenvolvimento dirigido a planos

- É baseado em estágios de desenvolvimento separado, com produtos a serem produzidos em cada um desses estágios planejados antecipadamente.
- Iteração incremental é possível no modelo cascata – dirigido a planos.
- Iterações ocorrem dentro das atividades.

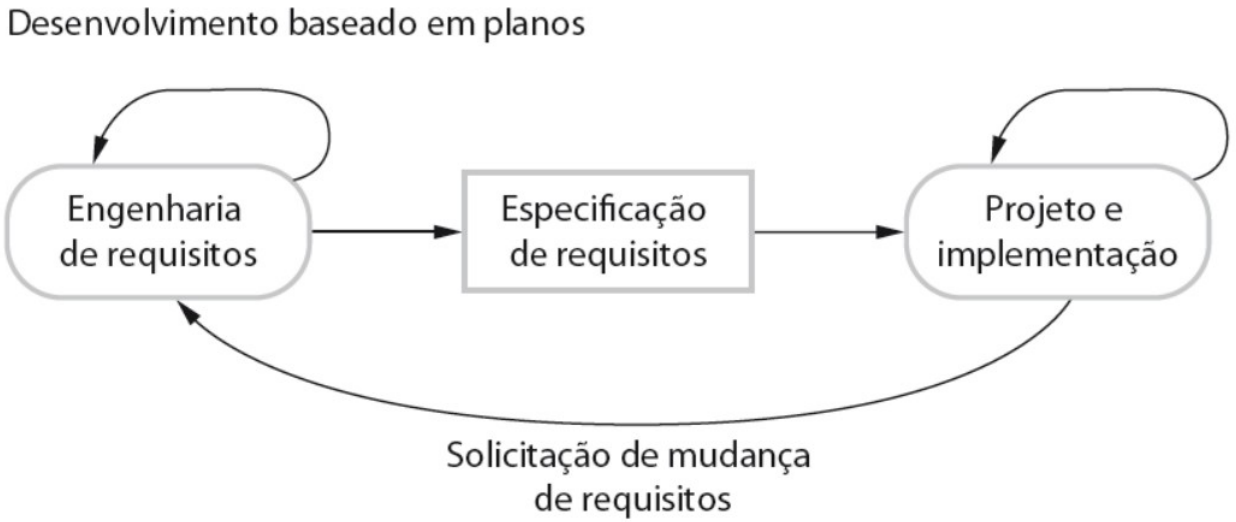
Modelo Cascata

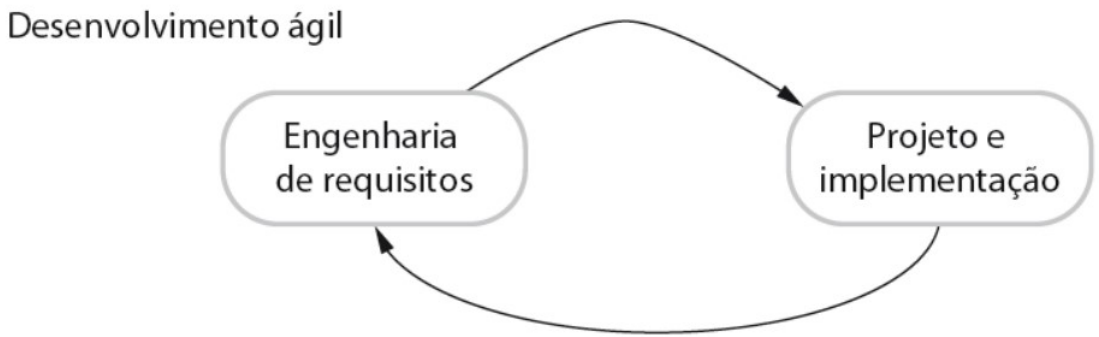


Desenvolvimento ágil

- Especificação, projeto, implementação e teste são intercalados e os produtos do processo de desenvolvimento são decididos através de um processo de negociação, durante o processo de desenvolvimento dos software.

Especificações dirigida a planos e ágil





Sommerville, 2011

Quando **usar** os métodos ágeis?

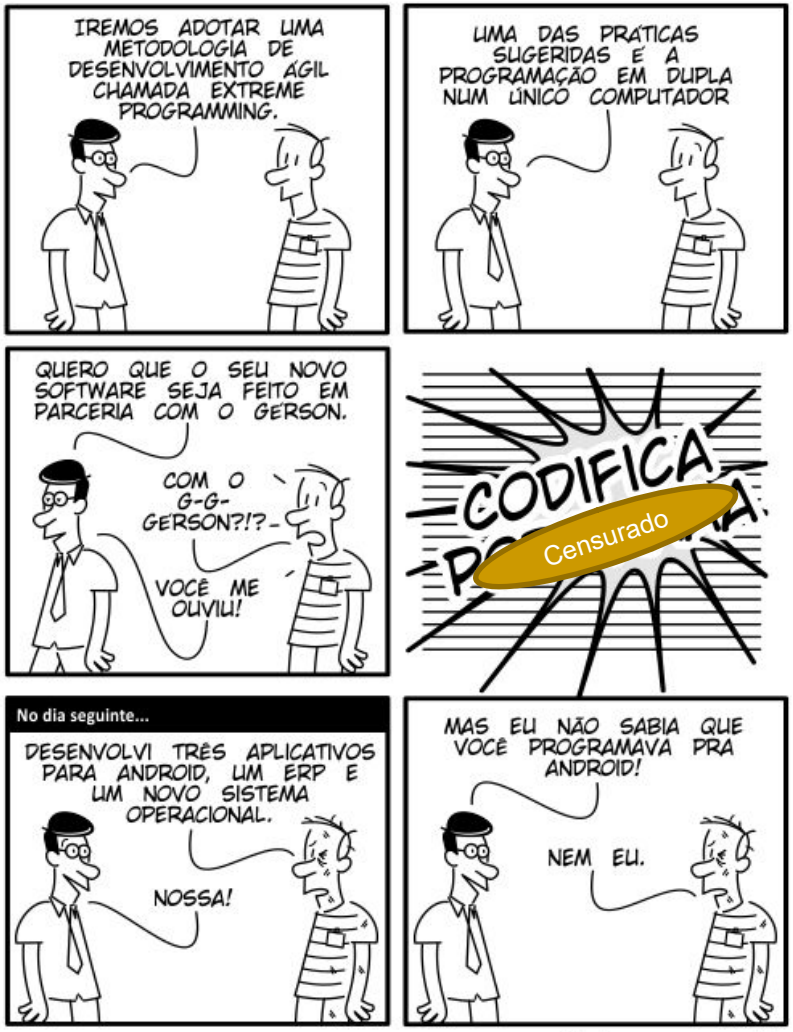
- A análise mais simples e básica indica, em geral, para:
  - Produtos novos.
  - Pequeno/médio porte.
  - Existe o compromisso do cliente se envolver no processo de desenvolvimento.
  - Não há muitas regras/regulamentos externos que afetam o software.
  - Equipes pequenas/integradas.

Quando **evitar** os métodos ágeis?

- Sistemas grandes e/ou complexos.
- Sistemas críticos
  - Os requisitos devem ser completamente especificado.
- Equipes grandes e/ou distribuídas.
- Cliente não está comprometido
  - Entrega uma especificação e apenas quer receber o software completo depois de um tempo

Programação Extrema (XP)

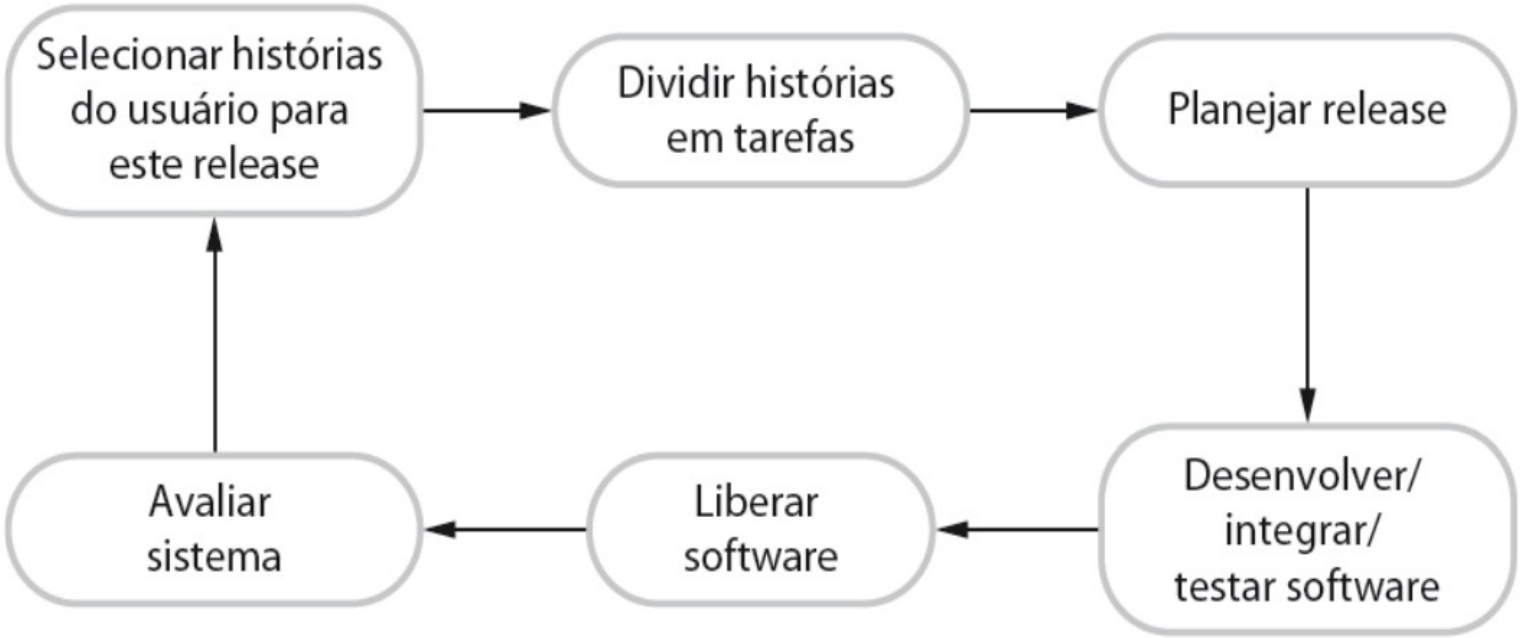
No dia seguinte...



Programação Extrema (XP)

- Proposta a partir de boas práticas de desenvolvimento incremental
- Propõe o envolvimento do cliente ao extremo
  - O cliente (ou seu representante) deve estar disponível durante todo o desenvolvimento
- Programadores trabalham em pares

Ciclo de uma release no XP



Sommerville, 2011

Práticas do XP

- Planejamento incremental
- Pequenos releases
- Projetos simples
- Desenvolvimento test-first
- Refatoração
- Programação em pares
- Propriedade coletiva
- Integração contínua
- Ritmo sustentável
- Cliente no local

Teste em XP

- As principais características dos testes em XP são:
  - Desenvolvimento *test-first*;
  - Desenvolvimento de teste incremental a partir do cenários;
  - Envolvimento dos usuários no desenvolvimento de testes e validação;
  - Uso de *frameworks* de testes automatizados.

Programação em pares

- Trabalham juntos na mesma estação
  - É criada de maneira dinâmica, todos os membros da equipe trabalham uns com os outros
- Propriedade e responsabilidade coletiva
  - Programação sem ego
- Processo informal de revisão
  - Processo de inspeção mais barato do que inspeções formais de programa

Programação em pares ...

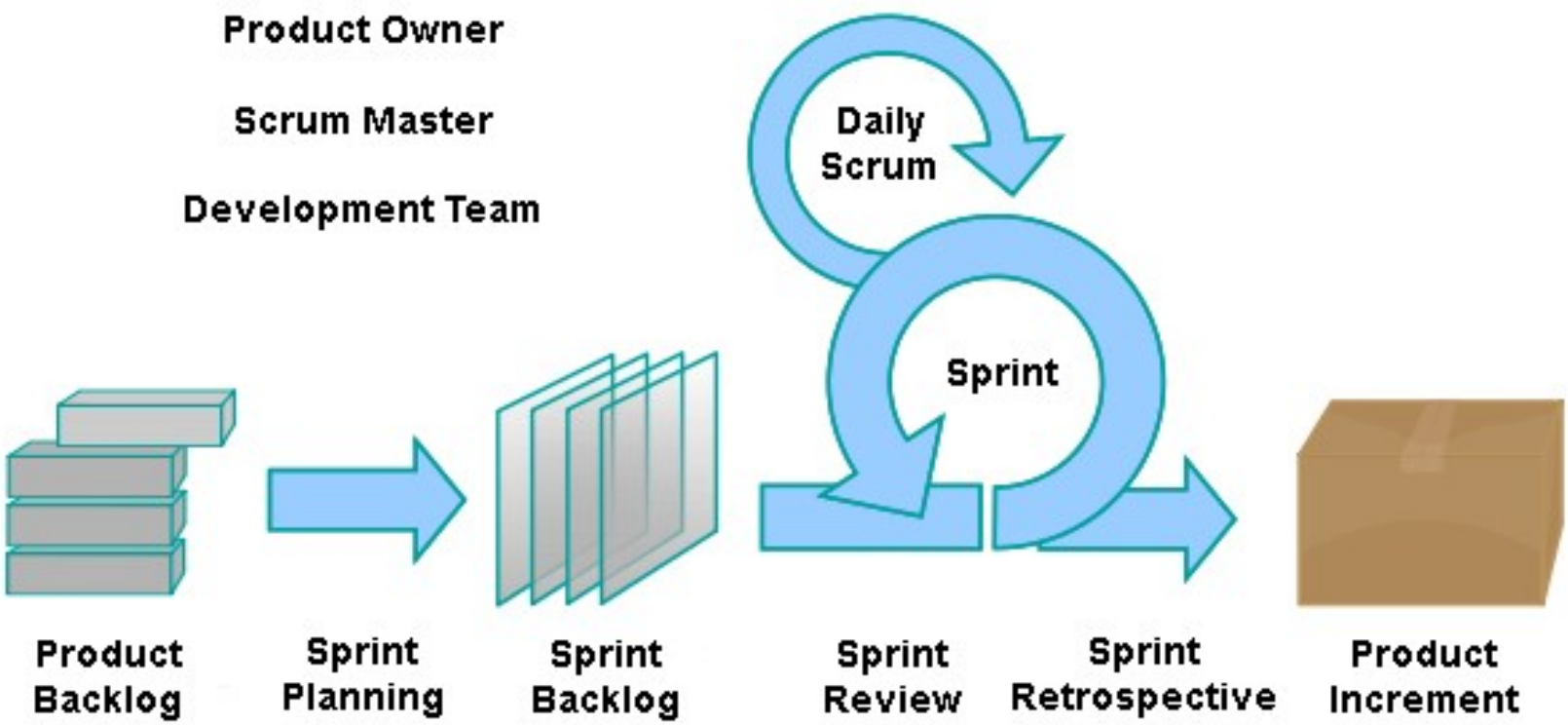
- Apoio à refatoração
  - Processo de melhoria de código
- Mais ganho para programadores menos experientes

Scrum

Método Ágil Scrum

- Método ágil que vem ganhando adeptos
- Scrum não é sigla, mas algumas empresas usam letras maiúsculas
  - SCRUM
- Objetivo: Oferecer uma forma de gerenciar projetos ágeis

Framework Scrum



Definição de papéis

- O papéis dos *stakeholders* caem em duas categorias **Porcos** e **Galinhas**

Product Owner

- O dono do produto é a pessoa que define os itens que compõem o *Product Backlog* e os prioriza nas reuniões de planejamento da *Sprint*.

Scrum Master (Facilitador)

- Líder da equipe
- Tem a função de remover impedimentos para que a equipe atinja os objetivos
- Garante que o processo está sendo usado e impõe a aplicação de regras
- Pode ser exercido por: gerente de projeto, ou um líder técnico, ou qualquer pessoa da equipe

Development Team

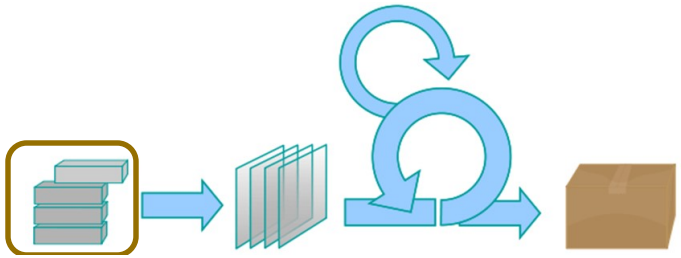
- É a equipe de desenvolvimento
- Responsável por entregar o produto
- Formado tipicamente por 5 a 9 pessoas

Stakeholders

- Representantes do Cliente
  - Pessoas que criam o ambiente para implantação do produto na organização
- Outros *Stakeholders*
  - Representam as várias pessoas envolvidas com o projeto
  - Podem ser clientes ou fornecedores

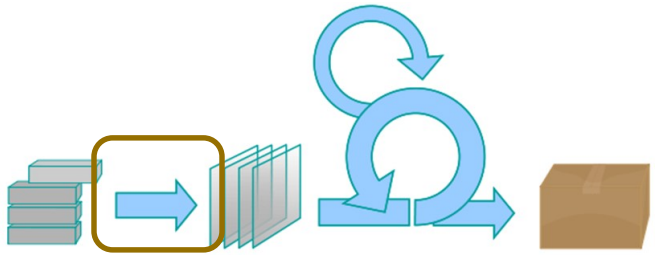
Product Backlog

- É uma lista contendo toda as funcionalidades esperada para um produto.
- O conteúdo desta lista é definido pelo dono do produto.
- Pode ser tarefas técnicas ou diretamente relacionadas as funcionalidades.



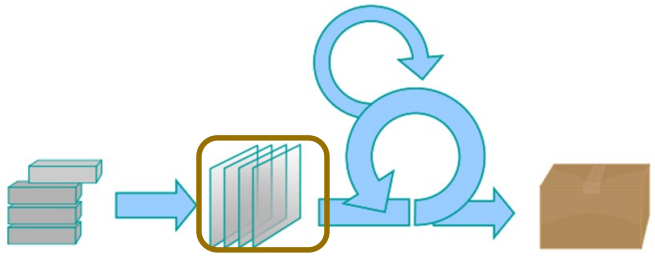
Sprint Planning

- Os presentes: dono do produto, *Scrum Master*, equipe de desenvolvimento e o representante da gerencia ou do cliente.
- Priorizar as funcionalidades.
- Quebrar as funcionalidades em tarefas técnicas.
- Define objetivo do *Sprint.*



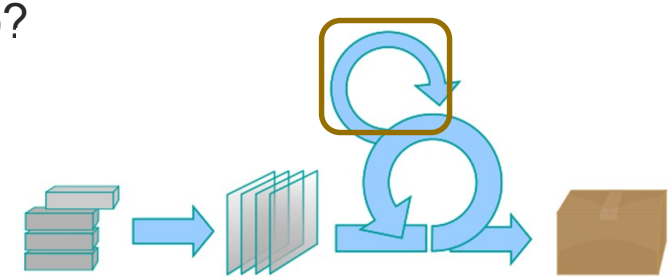
Sprint Backlog

- É uma lista de tarefas que a equipe de desenvolvimento se compromete a fazer em uma Sprint.
- Os itens são extraídos do *Product Backlog* com base nas prioridades definidas pelo dono do produto.



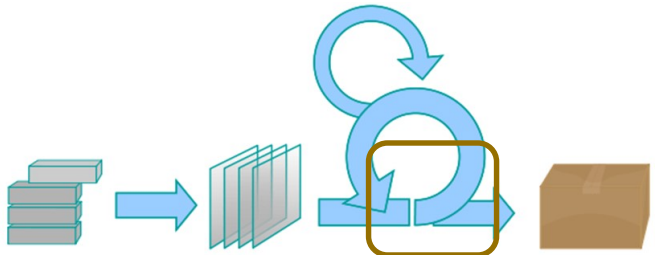
Daily Scrum

- Reuniões curtas e diária com a equipe de desenvolvimento, no mesmo local e horário.
- Objetivo: Disseminar o conhecimento, identificar impedimentos, priorizar as tarefas do dia.
- Perguntas básicas:
  - O que você fez ontem?
  - O que você fará hoje?
  - Há algum impedimento no seu caminho?



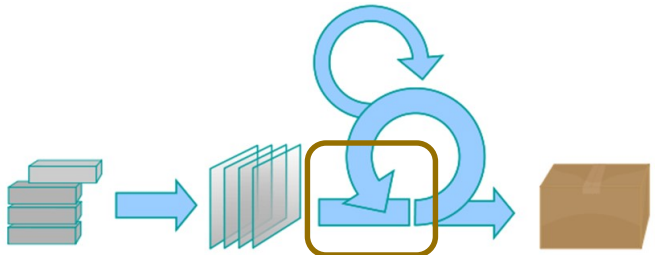
Sprint Retrospective

- Ocorre no final da *Sprint*
- Fazer o levantamento de tudo que foi positivo e negativo durante a *Sprint*
- Melhorar a próxima *Sprint*

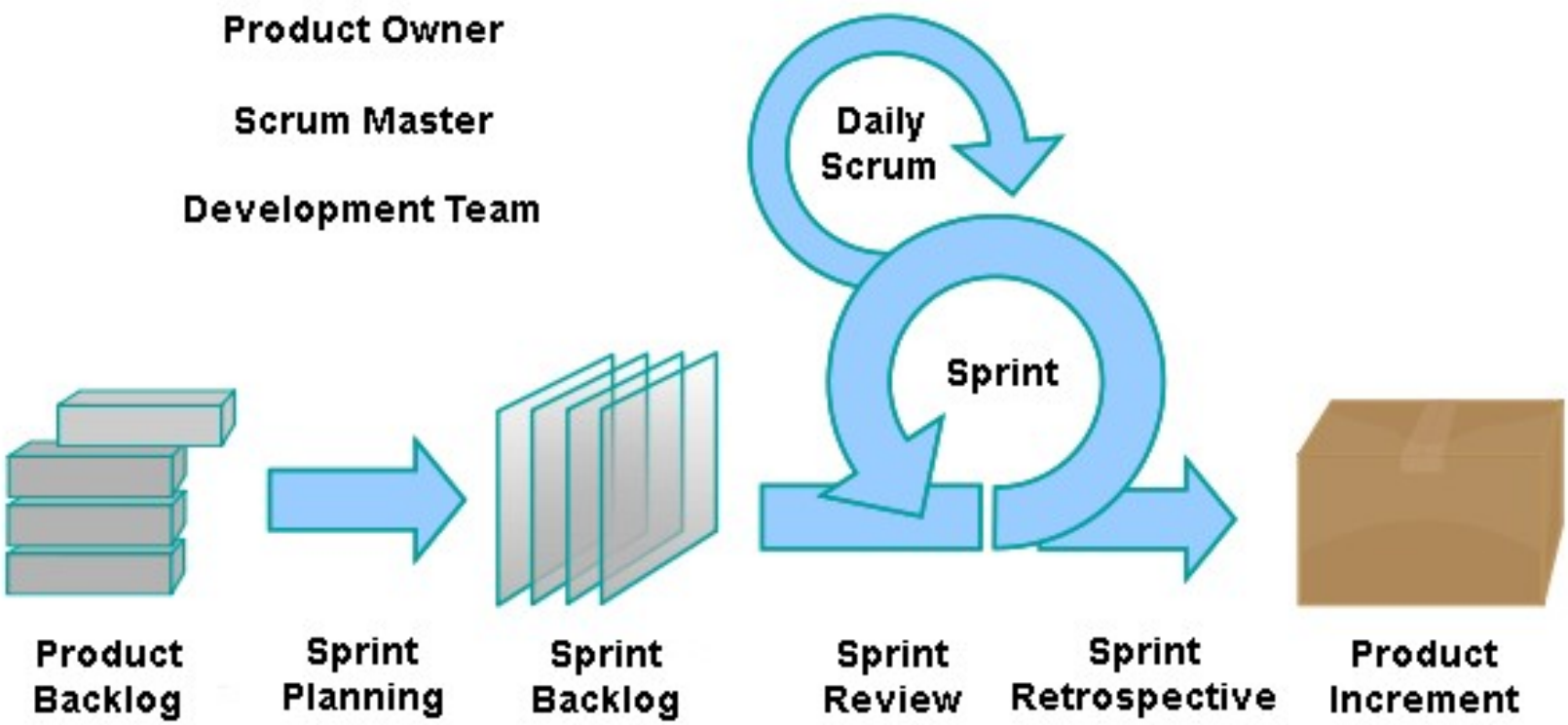


Sprint Review

- Os presentes: dono do produto, Scrum Master, equipe de desenvolvimento, gerência, clientes e engenheiros de outros projetos.
- Avalia se o planejamento e o objetivo da *Sprint* foi alcançado.
- Apresentar o resultado do *Sprint* aos stakeholders (demo)



Framework Scrum



O antes e o depois...

Bibliografia

- Ian Sommerville. **Engenharia de Software**, 9ª Edição. Pearson Education, 2011.
  - Capítulo 3 Desenvolvimento ágil de software
- Manifesto Ágil
  - http://agilemanifesto.org/

Princípios dos métodos ágeis

Princípios	Descrição
Envolvimento do cliente	Os clientes devem estar intimamente envolvidos no processo de desenvolvimento. Seu papel é fornecer e priorizar novos requisitos do sistema e avaliar suas iterações.
Entrega incremental	O software é desenvolvido em incrementos com o cliente, especificando os requisitos para serem incluídos em cada um.
Pessoas, não processos	As habilidades da equipe de desenvolvimento devem ser reconhecidas e exploradas. Membros da equipe devem desenvolver suas próprias maneiras de trabalhar, sem processos prescritivos.
Aceitar as mudanças	Deve-se ter em mente que os requisitos do sistema vão mudar. Por isso, projete o sistema de maneira a acomodar essas mudanças.
Manter a simplicidade	Focalize a simplicidade, tanto do software a ser desenvolvido quanto do processo de desenvolvimento. Sempre que possível, trabalhe ativamente para eliminar a complexidade do sistema.

Sommerville, 2011

Práticas do XP

Princípio ou prática	Descrição
Planejamento incremental	Os requisitos são gravados em cartões de história e as histórias que serão incluídas em um release são determinadas pelo tempo disponível e sua relativa prioridade. Os desenvolvedores dividem essas histórias em 'Tarefas'. Veja os quadros 3.1 e 3.2.
Pequenos *releases*	Em primeiro lugar, desenvolve-se um conjunto mínimo de funcionalidades útil, que fornece o valor do negócio. *Releases* do sistema são frequentes e gradualmente adicionam funcionalidade ao primeiro *release*.
Projeto simples	Cada projeto é realizado para atender às necessidades atuais, e nada mais.
Desenvolvimento *test-first*	Um *framework* de testes iniciais automatizados é usado para escrever os testes para uma nova funcionalidade antes que a funcionalidade em si seja implementada.
Refatoração	Todos os desenvolvedores devem refatorar o código continuamente assim que encontrarem melhorias de código. Isso mantém o código simples e manutenível.

Sommerville, 2011

Práticas do XP (...)

Princípio ou prática	Descrição
Programação em pares	Os desenvolvedores trabalham em pares, verificando o trabalho dos outros e prestando apoio para um bom trabalho sempre.
Propriedade coletiva	Os pares de desenvolvedores trabalham em todas as áreas do sistema, de modo que não se desenvolvam ilhas de *expertise*. Todos os conhecimentos e todos os desenvolvedores assumem responsabilidade por todo o código. Qualquer um pode mudar qualquer coisa.
Integração contínua	Assim que o trabalho em uma tarefa é concluído, ele é integrado ao sistema como um todo. Após essa integração, todos os testes de unidade do sistema devem passar.
Ritmo sustentável	Grandes quantidades de horas-extra não são consideradas aceitáveis, pois o resultado final, muitas vezes, é a redução da qualidade do código e da produtividade a médio prazo.
Cliente no local	Um representante do usuário final do sistema (o cliente) deve estar disponível todo o tempo à equipe de XP. Em um processo de Extreme Programming, o cliente é um membro da equipe de desenvolvimento e é responsável por levar a ela os requisitos de sistema para implementação.

Sommerville, 2011