Engenharia de Software

Aula4: Exemplos de processos (Processo Unificado e processos ágeis)

Dra. Ana Patrícia F. Magalhães Mascarenhas

anapatriciamagalhaes@gmail.com

PLANO DE AULA

- Objetivo
  - Apresentar exemplos de processos de desenvolvimento de software
    - o processo unificado
    - Processo ágil SCRUM
- Bibliografia básica
  - SOMMERVILLE, I. Engenharia de Software. 9a edição. **Capítulos 2 e 3**. Pearson Addison Wesley. 2011.
  - PRESSMAN, R. , MAXIM, B. Engenharia de Software, **Capítulos 4 e 5**, 8th edição.
  - www.desenvolvimentoagil.com.br
  - www.manifestoagil.com.br/
  - metodologiaagil.com/
  - www.culturaagil.com.br/o-que-sao-metodos-ageis/

Processo Unificado (PU)

Processo Unificado

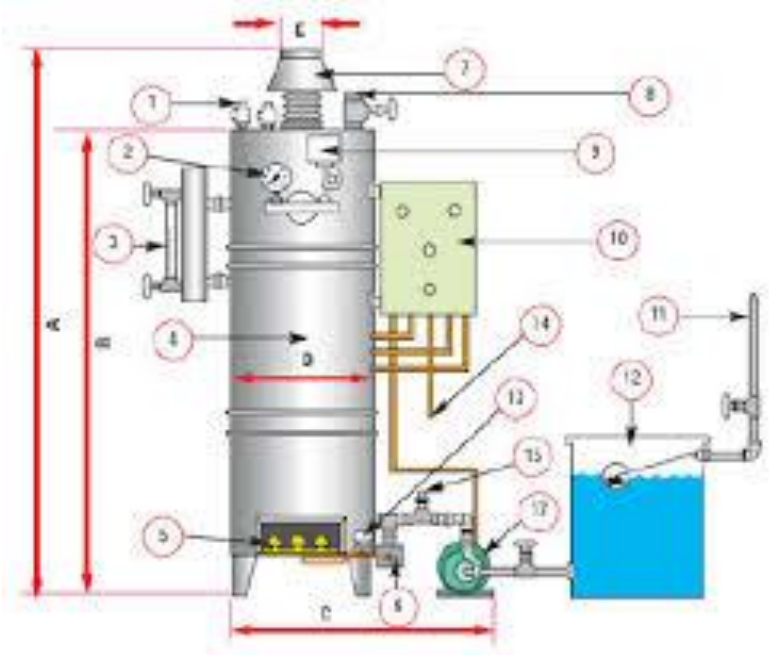
- Processo iterativo e incremental para o desenvolvimento de sistemas orientados a objetos
- Flexível e aberto, incentiva a inclusão de práticas interessantes de outros métodos iterativos a exemplo do Extreme Programming (XP) e Scrum
- Framework genérico que pode ser adaptado a uma grande classe de sistemas

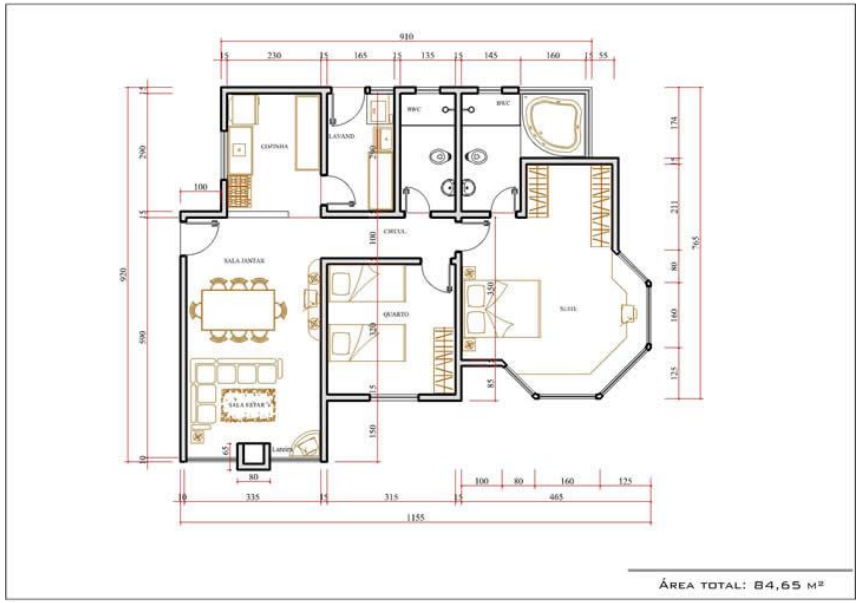
Práticas importantes adotadas pelo RUP

- Modelagem visual
- Iterativo e incremental
- Dirigido por casos de uso
- Centrado na arquitetura

Modelagem Visual

- Por que modelamos?
  - A modelagem é uma estratégia usada em diversas engenharias





  - **Construímos modelos para compreender melhor o sistema que estamos desenvolvendo**

Modelos

- Um modelo é uma visão simplificada do sistema. Mostra a essência do sistema em uma visão particular e esconde detalhes não essenciais.

- Importante para
  - Aumentar o entendimento de sistemas complexos
  - Explorar e comparar diferentes projetos
  - Formar uma base para a implementação
  - Capturar os requisitos com precisão
  - Comunicar as decisões de forma não ambígua

Desenvolvimento iterativo e incremental

- Desenvolvimento realizado em iterações que resultam em incrementos de novas funcionalidades.
- Cada incremento oferece valor para o cliente e é uma evolução da versão anterior

Requisitos do projeto
R1 R2 R3 R4 R5
R6 R7

Iterações planejadas
It1: R1, R4, R5
It2: R2, R7
It3: R3, R6

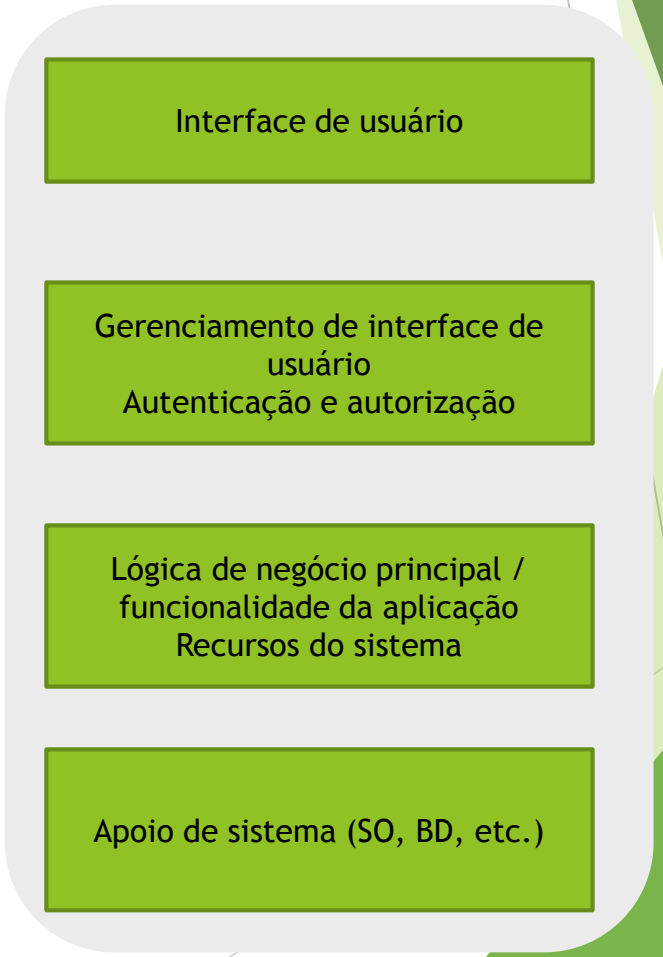
Desenvolvimento dirigido por caso de uso

- O caso de uso é a base para o desenvolvimento ao longo do processo
  - Especificam os requisitos
  - São utilizados no planejamento das iterações
  - Servem de condutor para o projeto
  - São utilizados para os testes

Desenvolvimento centrado na arquitetura

- A arquitetura mostra uma descrição compacta e gerenciável de como um sistema é organizado e como os componentes interoperam.
  - A arquitetura define a organização dos componentes que integram o sistema
  - A arquitetura mostra uma representação do sistema em alto nível, não é rica em detalhes.
    - Bom para discutir o sistema como um todo

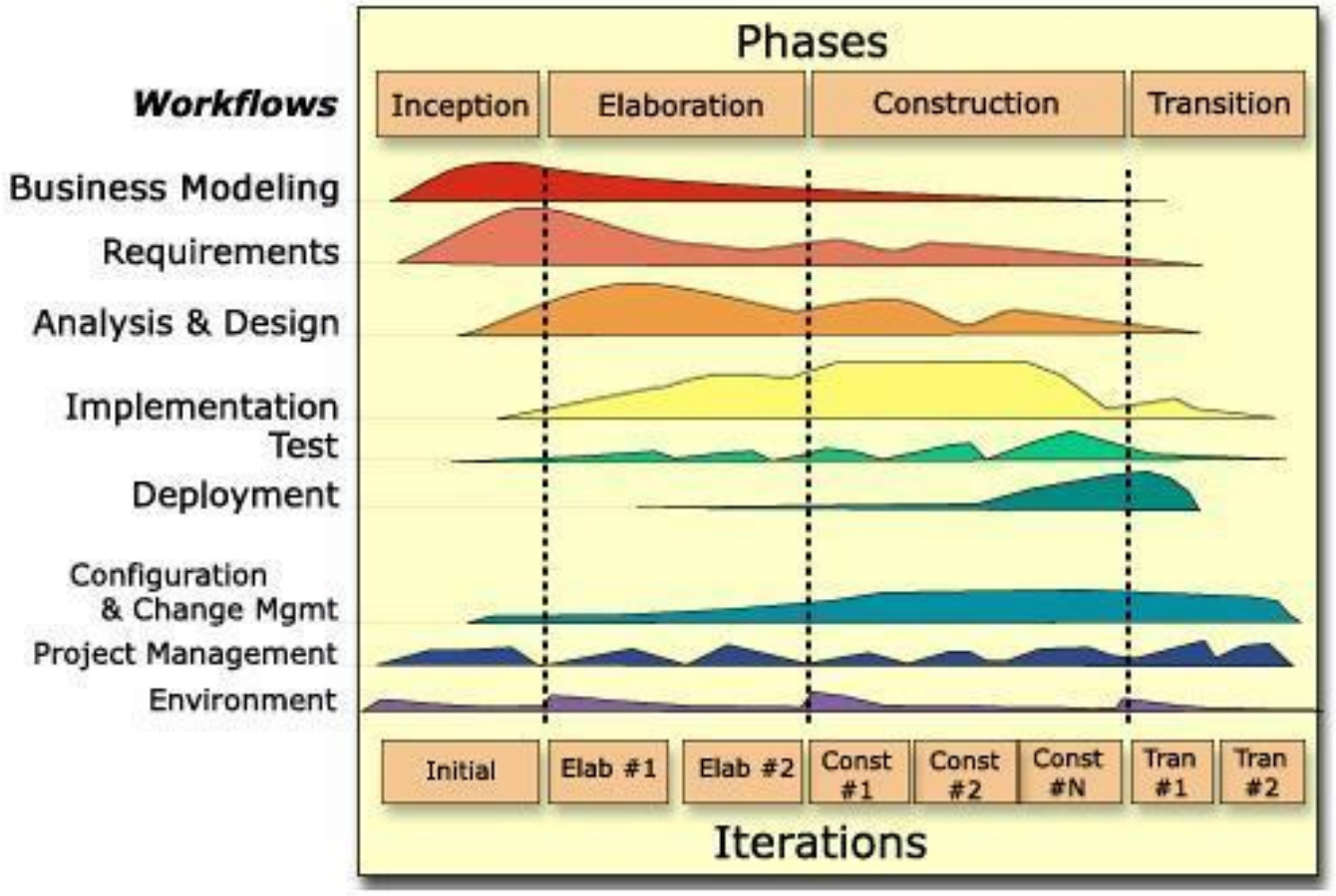
Arquitetura genérica em camadas



Estrutura do processo RUP (realização mais conhecida do PU)

- Bidimensional
  - a dimensão horizontal representando o tempo (estrutura dinâmica)
  - a dimensão vertical representando os fluxos de trabalho, agrupando atividades logicamente relacionadas (estrutura estática)

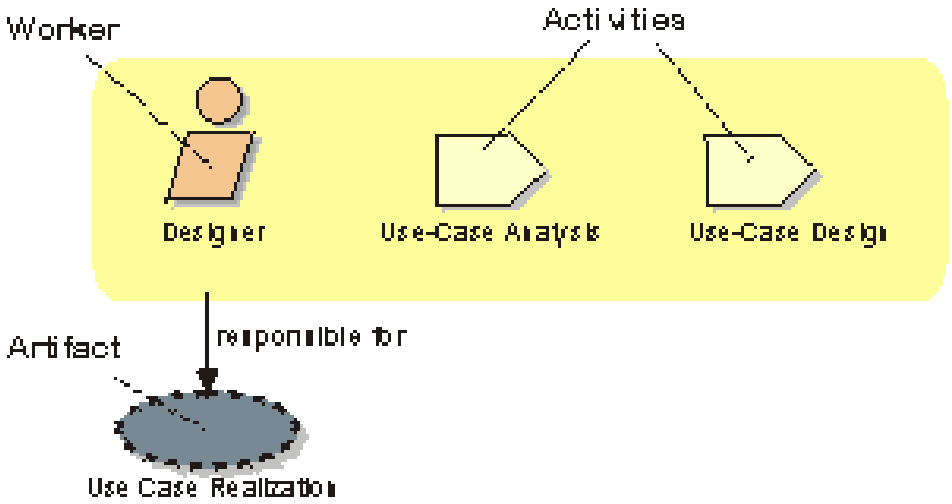
Estrutura Bidimensional



Fonte: Rational

Estrutura estática

- Um processo descreve *quem* estará fazendo o *que*, *como* e *quando*
  - *workers (roles)*: o *quem*
  - *activities:* o *como*
  - *artifacts:* o *que*
  - *workflow:* o *quando*



Fonte: Rational

Workers (papéis)

- Um papel define o comportamento e as responsabilidades de um indivíduo ou de um grupo de indivíduos que atuam juntos como uma equipe
- O comportamento é expressado em termos das atividades que o papel desempenha
- Cada papel é associado a um conjunto de atividades coesivas.

Exemplos de papéis

- Worker:System Analist
  - O analista de sistemas lidera e coordena a elicitação de requisitos e modelagem de casos de uso
- Worker:Tester
  - O tester é responsável por executar os testes, incluindo testes de setup e execução

Atividades (Activities)

- Papéis têm atividades, que definem o trabalho que eles deverão realizar
- Uma atividade é uma unidade de trabalho que uma pessoa exercendo um papel pode realizar e que produz um resultado significativo no contexto do projeto
- Uma atividade tem uma finalidade clara, usualmente expressa como a crição ou atualização de algum artefato, tal como um modelo, uma classe ou um plano

Exemplo de atividades

- Atividade: Planejar uma iteração:
  - Worker: Project Manager
- Atividade: Encontrar casos de uso e atores:
  - Worker: System Analyst
- Atividade: Rever o projeto:
  - Worker: Design Reviewer
- Atividade: Realizar teste de desenpneho
  - Worker: Performance Tester

Artefatos (Artifacts)

- Um artefato é um pedaço de informação que é produzido, modificado ou usado pelo processo
- Artefatos são produtos tangíveis
- Artefatos são usados como entradas por papéis para realizar uma atividade e são o resultado ou saídas destas atividades
- Em termos de orientação a objetos, as atividades são operações de um objeto ativo (o responsável), artefatos são os parâmetros destas atividades

Tipos de artefatos

- Artefatos podem assumir vários formatos:
  - Um modelo, tais como modelo de caso de uso ou modelo de projeto
  - Um elemento do modelo, tais como uma classe, um caso de uso ou um subsistema
  - Um documento, tais como documento de arquitetura de software
  - Código fonte
  - Executáveis

Fluxos de atividades (Workflows)

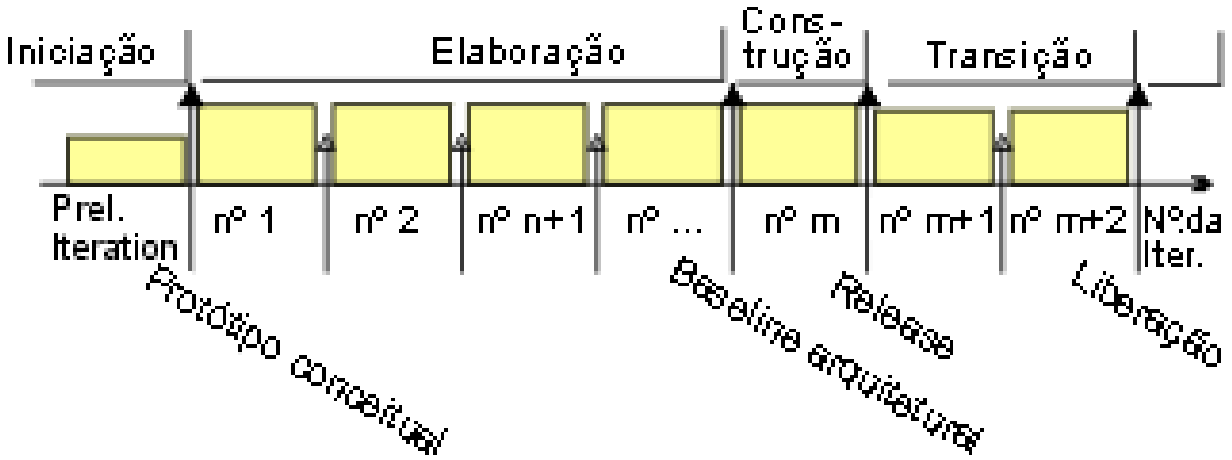
- A mera enumeração de papéis, atividades e artefatos não constituem um processo
- É necessária uma forma compreensível de descrever sequência de atividade que produzem algum resultado significativo e exibir as interações entre os papéis
- Um *workflow* é uma sequência de atividades que produzem um resultado de valor observável
- Em termos de UML, um *workflow* pode ser expresso como um digrama de atividade, sequência ou colaboração

Workflows

- UP define nove workflows, que representam o particionamento de todos os papéis e atividades em grupos lógicos
- Processos de engenharia
  - *Workflow* de Modelagem de Negócios
  - *Workflow* de Requisitos
  - *Workflow* de Análise e Projeto
  - *Workflow* de Implementação
  - *Workflow* de Teste
  - *Workflow* de Distribuição
- Workflows de Apoio
  - *Workflow* de Gerenciamento de Processos
  - *Workflow* de Gerenciamento de Configuração e Mudanças
  - *Workflow* de Ambiente

Estrutura dinâmica

- A estrutura dinâmica diz respeito ao modo como o processo se desenvolve através do tempo
- O processo é definido em *fases* e *milestones*
- Cada fase pode ter *n* iterações



Fonte: funpar.ufpr.br

Fases do UP

Fonte: https://pt.slideshare.net/adorepump/processo-unificadorup-presentation

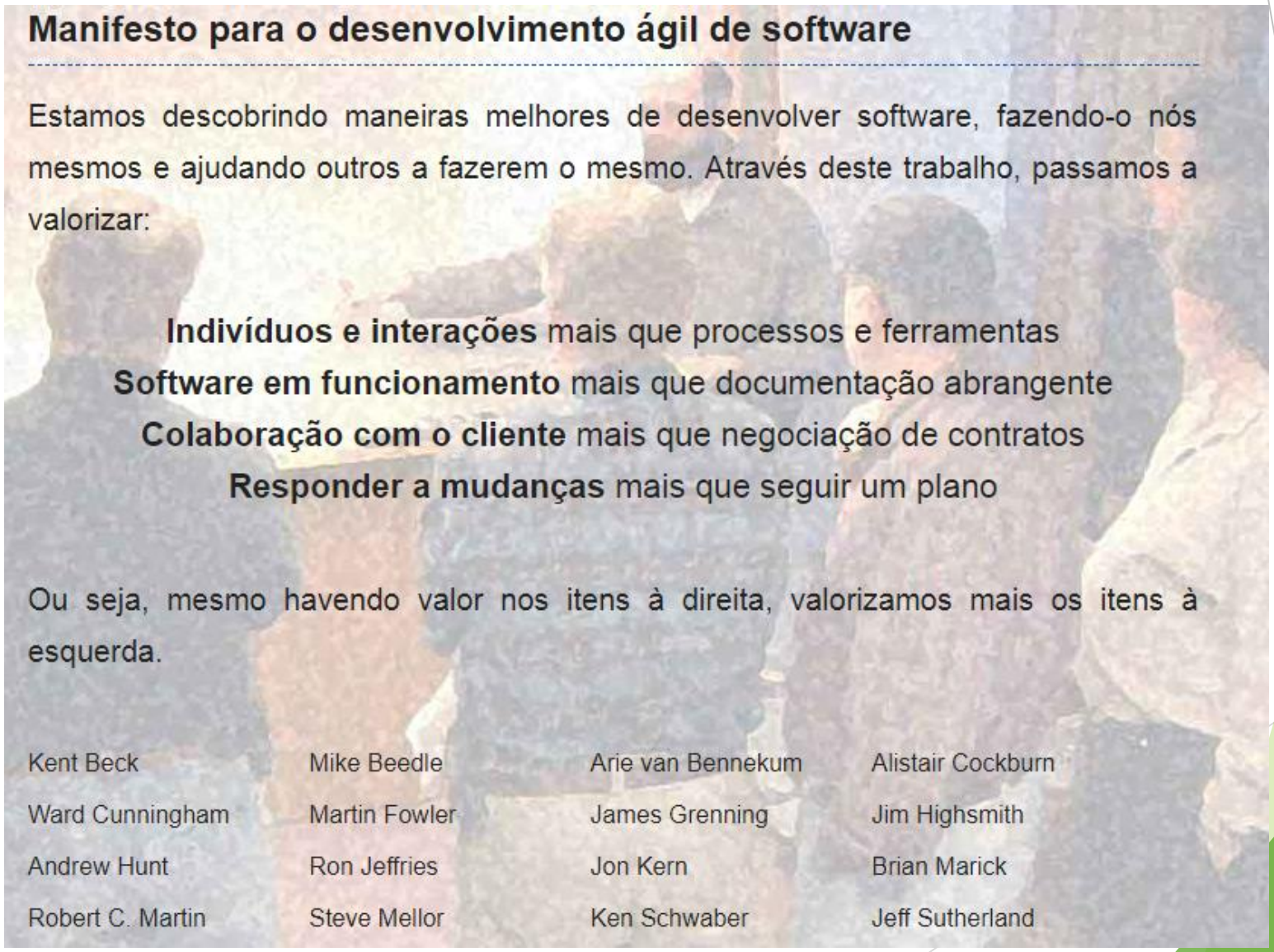
Processos Ágeis

O que é um processo ágil?

- É difícil prever quais requisitos vão persistir e quais serão alterados
- É difícil prever alterações nas prioridades dos usuários à medida que o projeto avança
- Em muitos tipos de software o projeto e a construção são intercalados, andam juntos, e é difícil prever o quanto de projeto será necessário fazer para implementar o software
- O planejamento de análise, projeto, construção e testes não são tão previsiveis

Como criar um processo capaz de administrar a imprevisibilidade?
O processo precisa ser **adaptável**

Manifesto Ágil: filosofia por trás dos métodos ágeis



Manifesto para o desenvolvimento ágil de software

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.

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

Os 12 princípios do manifesto ágil

1. Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor.

2. Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas.

3. Entregar software funcionando com freqüencia, na escala de semanas até meses, com preferência aos períodos mais curtos.

Os 12 princípios do manifesto ágil

4. Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto.
5. Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.
6. O Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara.
7. Software funcional é a medida primária de progresso.

Os 12 princípios do manifesto ágil

8. Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes.
9. Contínua atenção à excelência técnica e bom design, aumenta a agilidade.
10. Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.
11. As melhores arquiteturas, requisitos e designs emergem de. times auto organizáveis
12. Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.

Características fundamentais dos métodos ágeis (1)

- Processo de análise, projeto e implementação são concorrentes
  - Não há especificação detalhada de sistema
  - Documentação minimizada ou gerada automaticamente pelo ambiente de programação
  - Documento de requisitos define apenas as características mais importantes

Características fundamentais dos métodos ágeis (2)

- Interface geralmente desenvolvida usando um sistema de desenvolvimento interativo
  - Projeto da interface é criado rapidamente por meio de desenho de telas, etc.

Características fundamentais dos métodos ágeis (3)

- Sistema desenvolvido em uma série de incrementos
  - Todos os envolvidos com o sistema (*stakeholders*) participam da especificação e da avaliação de cada incremento
  - Alterações ou novos requisitos podem ser propostos para um incremento posterior

O que é agilidade então?

Agilidade é mais que resposta a mudança

É uma filosofia proposta no manifesto ágil

Incentiva à atitude da equipe

Visa entrega rápida de software operacional

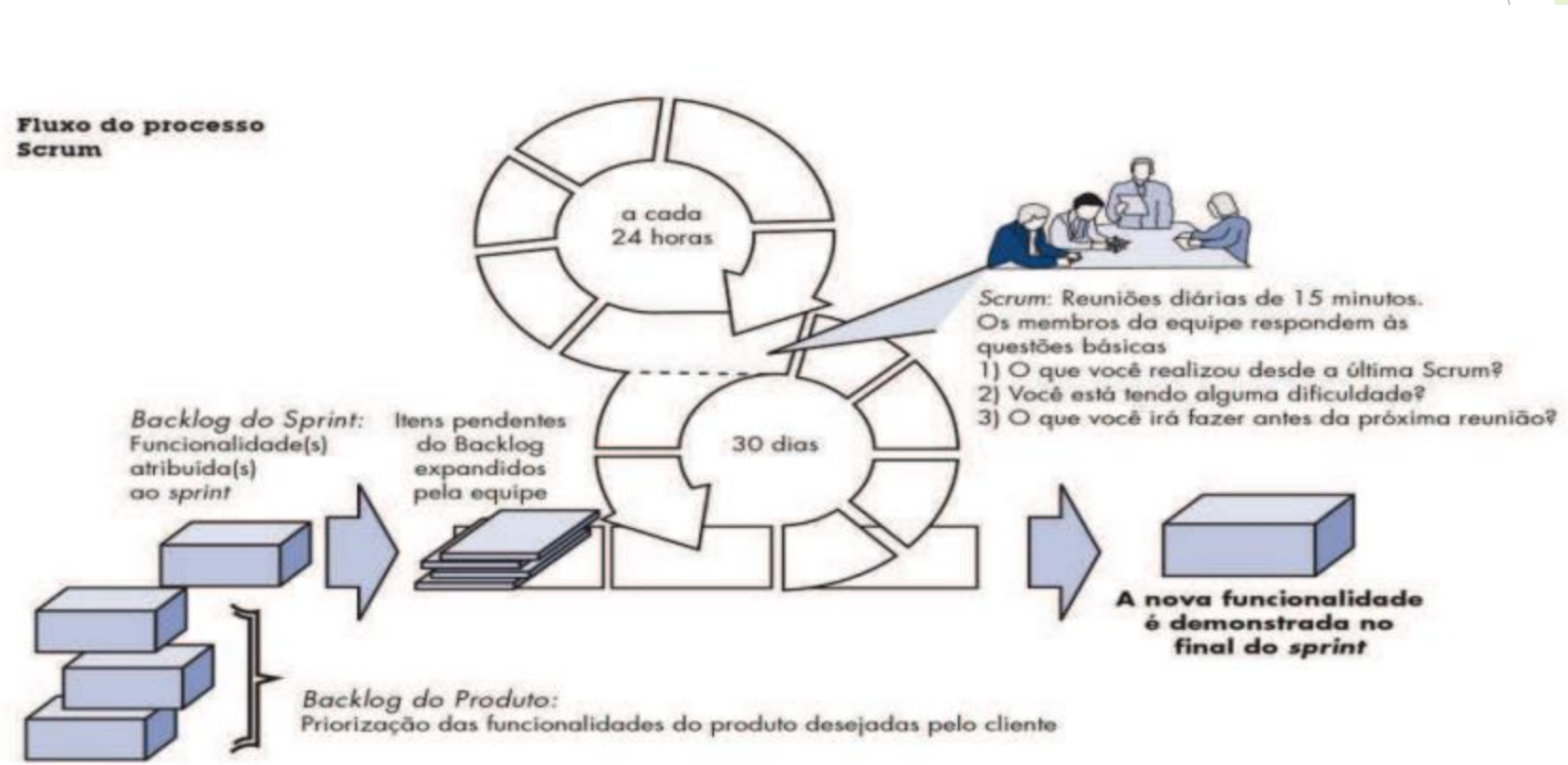
Diminui a importância de artefatos intermediários

Admite que Planejamento tem limites e que o plano deve ser flexível

Aceita o cliente como parte da equipe

SCRUM

- Criado por Jeff Sutherland e sua equipe nos anos 90
- Princípios coerentes com os métodos ágeis
- Indicado para projetos com prazos apertados, requisitos que mudam com frequência e que são críticos para o negócio

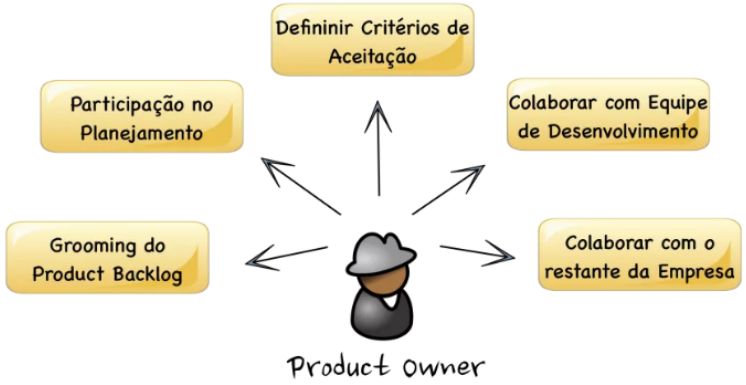


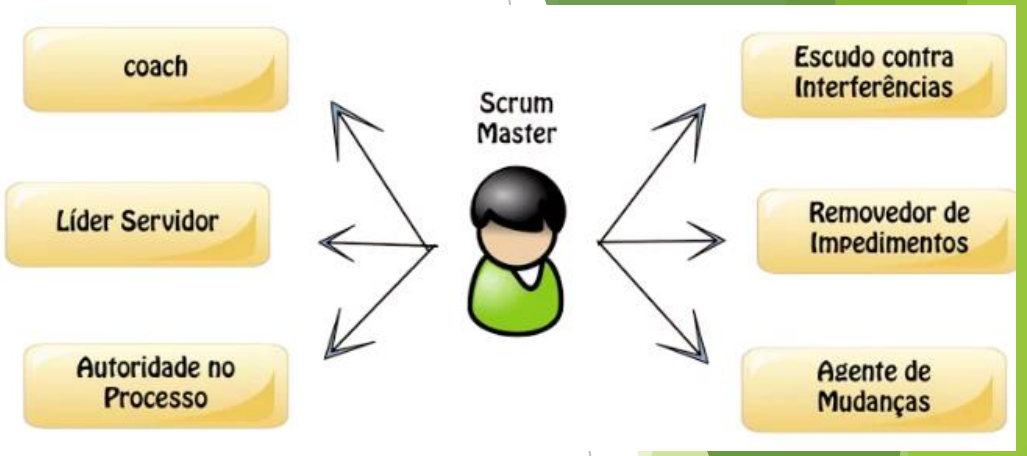
Fluxo do processo Scrum (Pressmman, 2016, p.78)

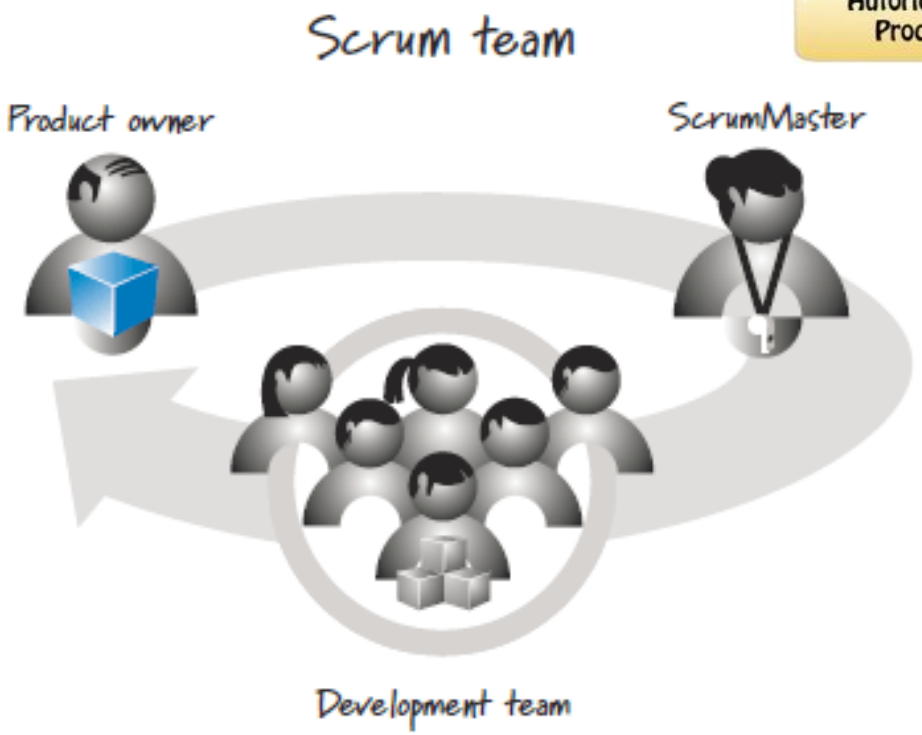
SCRUM

- Pre-Jogo
  - Planejamento
  - Arquitetura
  - Definição de pronto - DoD
- Jogo
  - Desenvolvimento
  - Sprints
  - Reunioes diárias
- Pos-Jogo
  - Fechamento
  - Retrospectiva

Papeis do SCRUM







Disponível em http://www.mindmaster.com.br/scrum/

Estórias do usuário

- Descrevem o que precisa ser desenvolvido, ou seja, quais os requisito do cliente
- Deve agregar valor para o cliente
- É o que chamamos no desenvolvimento tradicional de Caso de Uso
- Algumas estórias são chamas de Épico
  - Nível alto de abstração
  - Grandes
- Estórias podem estar agrupadas em temas

Backlog do produto (Product backlog)

- Lista com todas as funcionalidades do sistema a ser desenvolvido
  - Deve ter valor para o cliente
  - **Contém as *estórias* que devem ser desenvolvidas**
- Deve estar priorizado
- É definido pelo *ProductOwer* (PO)
- Novos itens podem ser adicionados ao longo do tempo
  - Com o uso do produto
  - Devido a mudanças no negócio e no ambiente
- As prioridades são sempre reavaliadas

As estórias formam o Product Backlog

As estórias que estão no topo da lista precisam estar mais bem detalhadas

Planejando uma Sprint – Reunião de planejamento 1

- Define o que vai ser feito na próxima Sprint
- O PO inicia a reunião com uma proposta de meta
  - Explica quais as estórias que ele priorizou de acordo com a meta
  - Equipe tira suas dúvida
- Realiza-se o *planning poker* para estimar o esforço
- Com base no resultado do *planning poker* se define o que vai ser colocado no *Sprint backlog*
- Leva normalmente 4 horas para uma Sprint de 30 dias
- Ao final da reunião a equipe decidiu junto com o PO o que vai ser entregue (*Sprint Backlog*)
- Muitas coisas são discutidas
  - Há problemas de tecnologia, fornecedor, recursos de pessoas
- Participam da reunião o PO, o ScrumMaster, a equipe e mais algum especialista da área, se for o caso

Planejando uma Sprint – Reunião de planejamento 2

- A equipe vai definir como vai realizar as estórias
- Permite que a equipe planeje seu trabalho para aquelas estórias
- As estórias são divididas em tarefas
  - Tarefas são identificadas
  - Tarefas são estimadas em horas (opcional)
  - Preferencialmente 1 dia de trabalho
  - Ex. criar a base de dados, criar uma tela
- O PO ou especialistas podem participar para tirar dúvidas (opcional)
- A equipe tem que começar o trabalho sem dúvidas

Sprint

- Iteração para desenvolvimento do produto
- Realizado tipicamente em 15 a 30 dias
- Quanto maior o risco menor deve ser a Sprint
- O membro da equipe pega uma tarefa e executa
- Pode ser necessário conversar com o PO ou um especialista
  - Devem estar disponíveis
- Discute impedimentos, algo externo ao projeto que faz o projeto parar
  - Ex. a equipe depende de um componente de um fornecedor que não chegou
- Alterações neste período não são permitidas
  - É possível incluir uma estória, mas não é recomendado
- Equipe trabalha em um ambiente de curto prazo, mas estável
- Uma release pode ser fruto de várias *sprints*
    - Uma Sprint leva a um produto potencialmente pronto

Reuniões de revisão (Demo)

- Realizada no final de cada Sprint
- Time se reúne com o PO para mostrar o que fizeram na Sprint
- O time mostra o que fizeram
- O PO aceita ou não baseada na avaliação de pronto
  - *Definition of Done (DoD)*
- Se não aceitar, volta para o product backlog para ser novamente priorizada
- O PO faz uma avaliação de se a meta da Sprint foi atingida
  - Com base no que foi aceito ou não
- Gera subsídios para a reunião de planejamento da próxima Sprint
- Tem duração de mais ou menos 4 horas para uma Sprint de 30 dias
- Importante a participação do PO dando retorno a equipe do trabalho realizado.

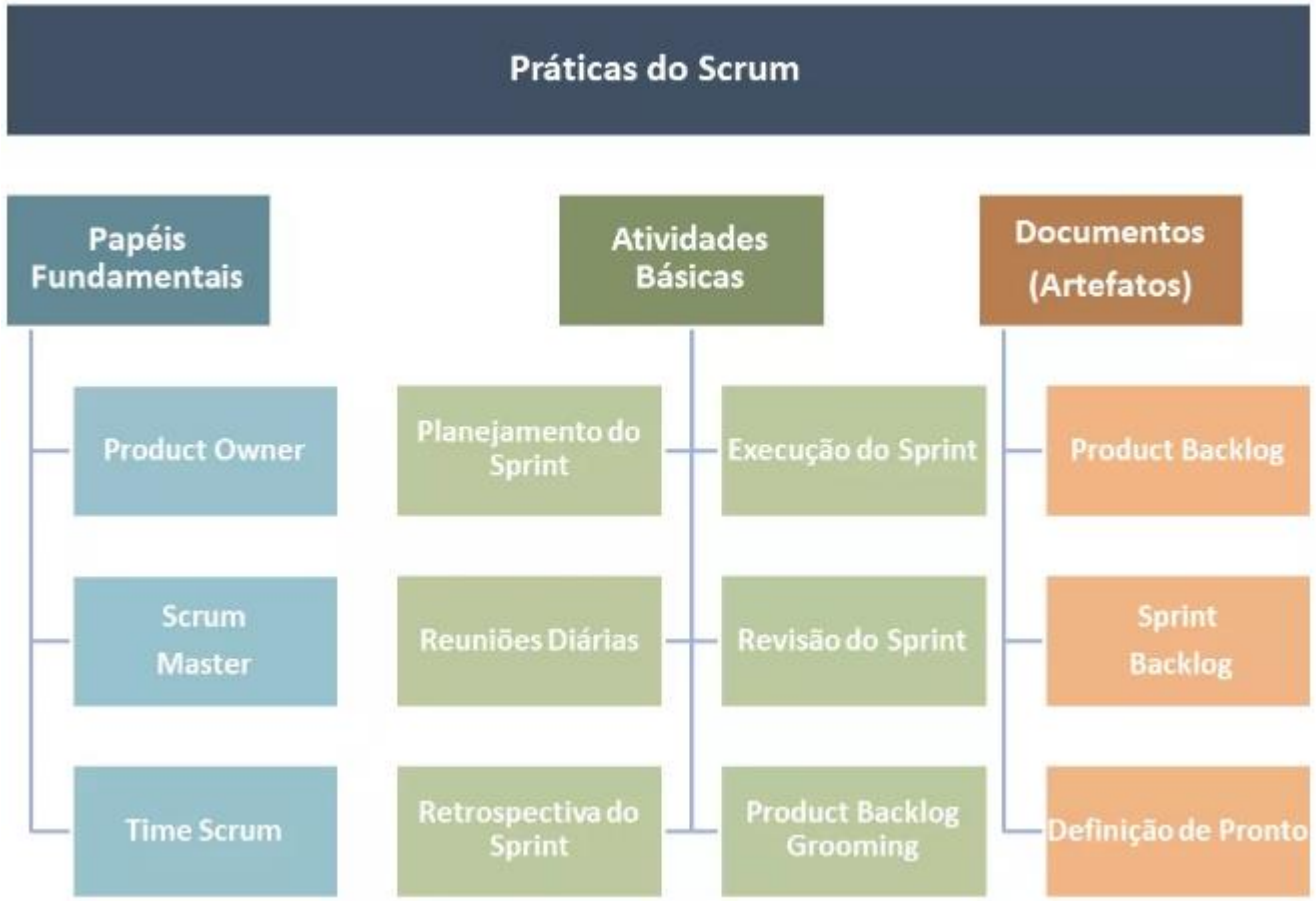
Product Backlog Grooming

- Product BackLog Grooming
  - Revisão dos itens do product backlog para garantir que são apropriados.
  - É uma revisão regular ou sobre demanda
    - Remove estórias que não são mais relevantes
    - Cria novas estórias a partir de novas necessidades
    - Revê as prioridades
    - Corrige as estimativas a partir de novas informações
    - Divide estórias de alta prioridade e que fazem muitas coisas para ajustá-las às novas iterações

Reuniões de Retrospectiva

- Última reunião
- Conduzida pela equipe
- O PO nunca vai a reunião
  - Só se for convidado, mas não é recomendado
- Equipe vai listar pontos positivos e negativos na execução da Sprint
- Cada ponto negativo deve ser discutido e definido uma solução
- A ideia é melhorar a qualidade do trabalho da equipe
  - Comunicação e colaboração
  - Definição das estórias
  - Como os testes estão sendo realizados

Resumindo



Disponível em http://www.mindmaster.com.br/scrum/

Exercícios

- Vamos pesquisar sobre o uso processos no desenvolvimento de software:
  - qual o modelo de processo mais usados?
  - existe alguma estatística sobre o uso de processos?
  - cenário de uso de métodos ágeis no Brasil
  - quais os métodos ágeis mais usados?
  - qual o porte das empresas?