# Engenharia de Software I
# Arquitetura de Software
# Processos Ágeis de Modelo de Software

Professor Doutor Flávio Miranda de Farias
Curso Tecnólogo Sistemas para Internet
Instituto Federal de Ciência e Tecnologia do Acre - IFAC
2024-2

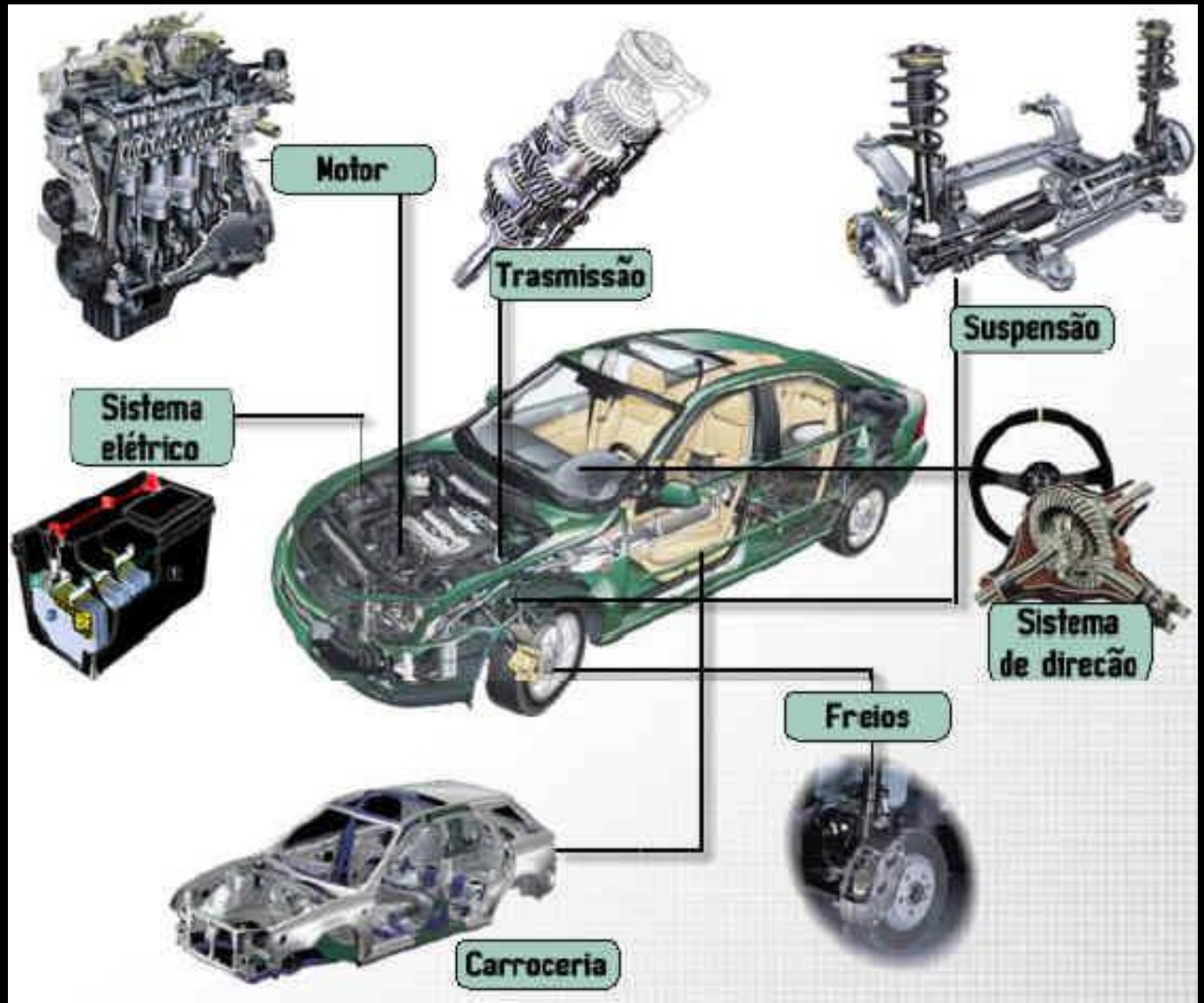
# Processos Ágeis de Modelo de Software

- **Principais processos de modelo Software.**
  - **Clássicos**
    - Sequencial Linear (cascata).
    - Prototipagem (rascunhos).
    - RAD (desenvolvido em partes).
    - Modelos Evolucionários (onde prevê a união das partes).
    - Espiral (foco em projetos de risco).
  - **Modernos**
    - Engenharia de Software baseada em componentes (reaproveitamento).
    - Processo Unificado (voltado a grandes projetos).
    - Programação Extrema (XP) (desburocratização).
    - Scrum, Kanban (gerenciamento de software).

# Conteúdo Programático.

- **Modernos**
  - Engenharia de Software baseada em componentes (reaproveitamento).
  - Processo Unificado (voltado a grandes projetos).
  - História dos métodos ágeis;
  - Programação Extrema (XP) (desburocratização).
  - Scrum, Kanban (gerenciamento de software).

# Processos de Software – Engenharia de Software Baseada em Componentes



# Processos de Software – Engenharia de Software Baseada em Componentes

- CBSE – *Component-Based Software Engineering*).
- Apoiada pela **orientação a objetos**.
  - Final da década de 90 como frustração dos projetistas quanto ao OO não conduzir ao reuso amplo. (detalhamento e classes demasiado).
  - Classes se adequadamente projetadas e implementadas, são reusáveis ao longo de diferentes aplicações e arquiteturas de sistemas baseados em computadores.

# Processos de Software – Engenharia de Software Baseada em Componentes

- Incorpora muitas das características do modelo espiral evolucionário.
- Compõe a aplicação de software a partir de componentes de software previamente preparados.
- Processo de definição, implementação e integração ou composição de componentes **INDEPENDENTES**, **não firmemente acoplados ao sistema.**

# Processos de Software – Engenharia de Software Baseada em Componentes

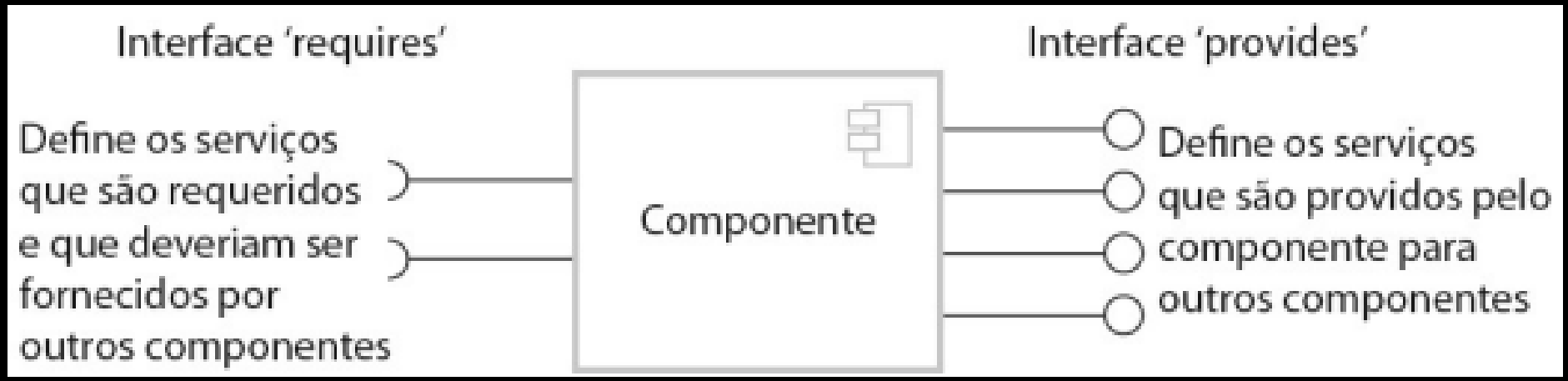
- “Um componente é uma unidade de composição com interfaces contratualmente especificadas e dependências de contexto explícitas somente. Um componente de software pode ser implantado independentemente e está sujeito à composição por terceiros.” (Szyperski, 2002).
- Componente:
  - Entidade executável independente.
  - Seus serviços estão disponíveis por meio de uma interface, por onde todas as suas interações ocorrem.

# Processos de Software – Engenharia de Software Baseada em Componentes

- Características de um componente:
  - Padronizado.
  - Independente.
  - Passível de composição.
  - Implantável.
  - Documentado.
- Interfaces de um componente:
  - ***Requires*** *(entradas)*: quais serviços devem ser fornecidos por outros componentes do sistema.
  - ***Provides*** *(saídas)*: serviço fornecido pelo componente.

Ver Exemplo

# Requires e Provides



# Processos de Software – Engenharia de Software Baseada em Componentes

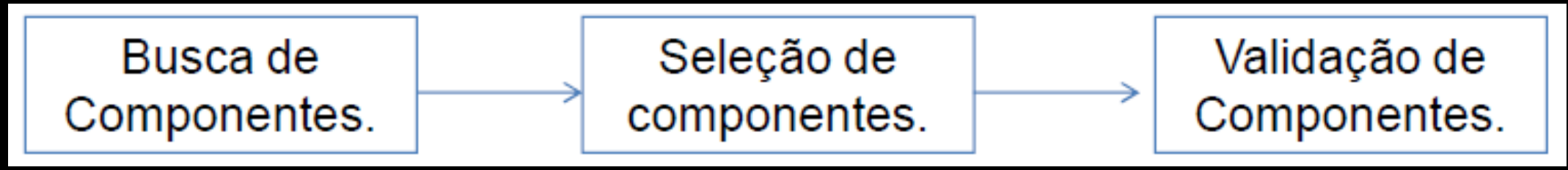
- Pontos essenciais:
  - Componentes independentes completamente especificados por suas interfaces.
  - Padrões de componentes que facilitam a integração de componentes.
  - Middleware que fornece apoio de software para integração de componentes.
  - Processo de desenvolvimento voltado à engenharia de software baseada em componentes.

# Processos de Software – Engenharia de Software Baseada em Componentes

- Componentes X Orientação a Objetos:
  - Componentes são entidades implantáveis.
  - Componentes não definem tipos.
  - Implementações de componentes são opacas.
  - Componentes são independentes de linguagem.
  - Componentes são padronizados.
- Modelos de componentes:
  - COM+.
  - EJB.
  - CORBA.

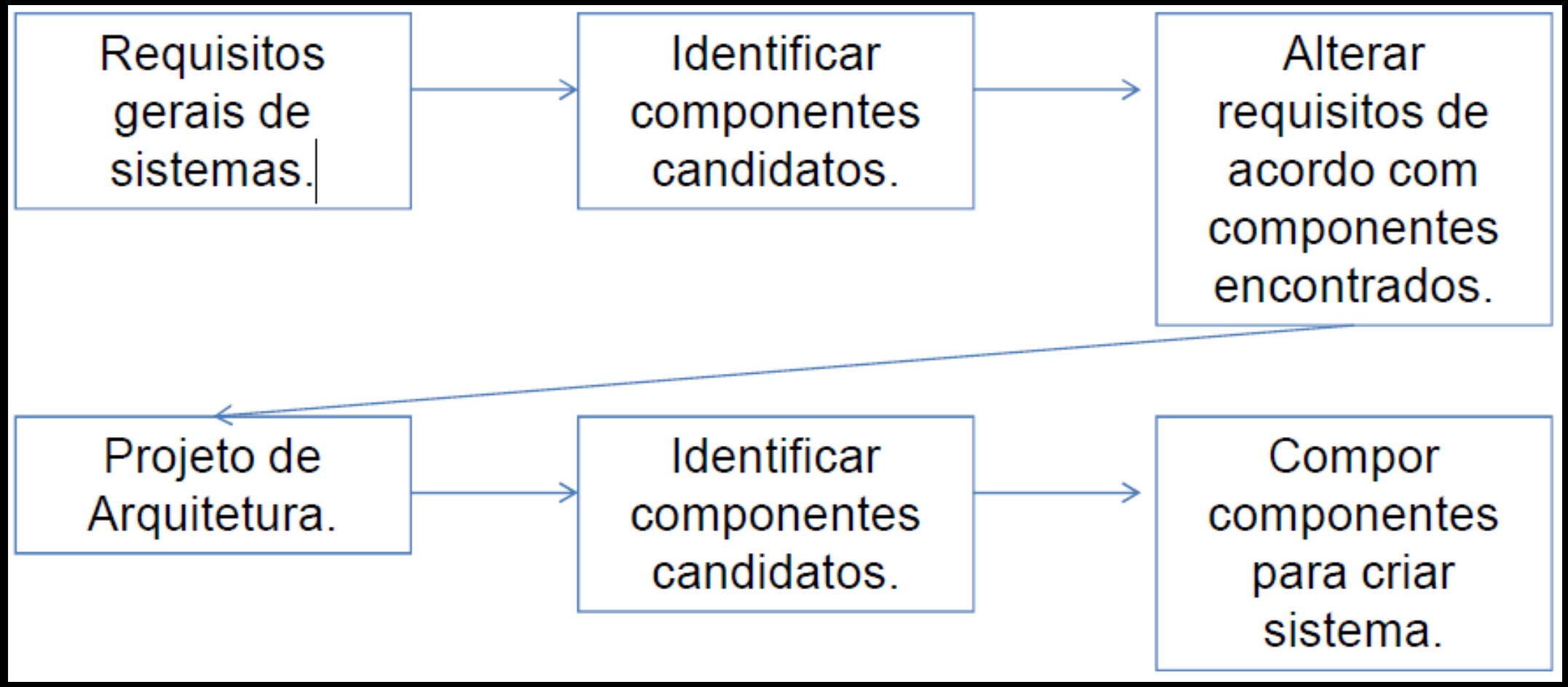
# Processos de Software – Engenharia de Software Baseada em Componentes

- O fato de um componente ser reusável depende de seu domínio de aplicação e funcionalidade. (Quanto mais genérico, mais reusável é um componente).
- Reusabilidade X usabilidade.
- Processos de CBSE.



# Processos de Software – Engenharia de Software Baseada em Componentes

- Processo de identificação de componente.



# Processos de Software – Engenharia de Software Baseada em Componentes

- A primeira iteração será composta pelas classes extraídas da biblioteca e novas classes construídas.
- O fluxo volta para ao espiral e tudo se repete.
- Tem como palavra chave o reuso.
- Reduz em até 70% a montagem de componentes.
- Reduz em até 84% o custo do projeto.
- Podemos dizer que o **Processo Unificado** foi originado da Engenharia de Software baseada em componentes.

# Conteúdo Programático.

- **Modernos**
  - Engenharia de Software baseada em componentes (reaproveitamento).
  - Processo Unificado (voltado a grandes projetos).
  - História dos métodos ágeis;
  - Programação Extrema (XP) (desburocratização).
  - Scrum, Kanban (gerenciamento de software).

# Processos de Software

## Processo Unificado (PU) – *Rational Unified Process* – (RUP)

# Processos de Software – Processo Unificado (PU) – *Rational Unified Process (*RUP)

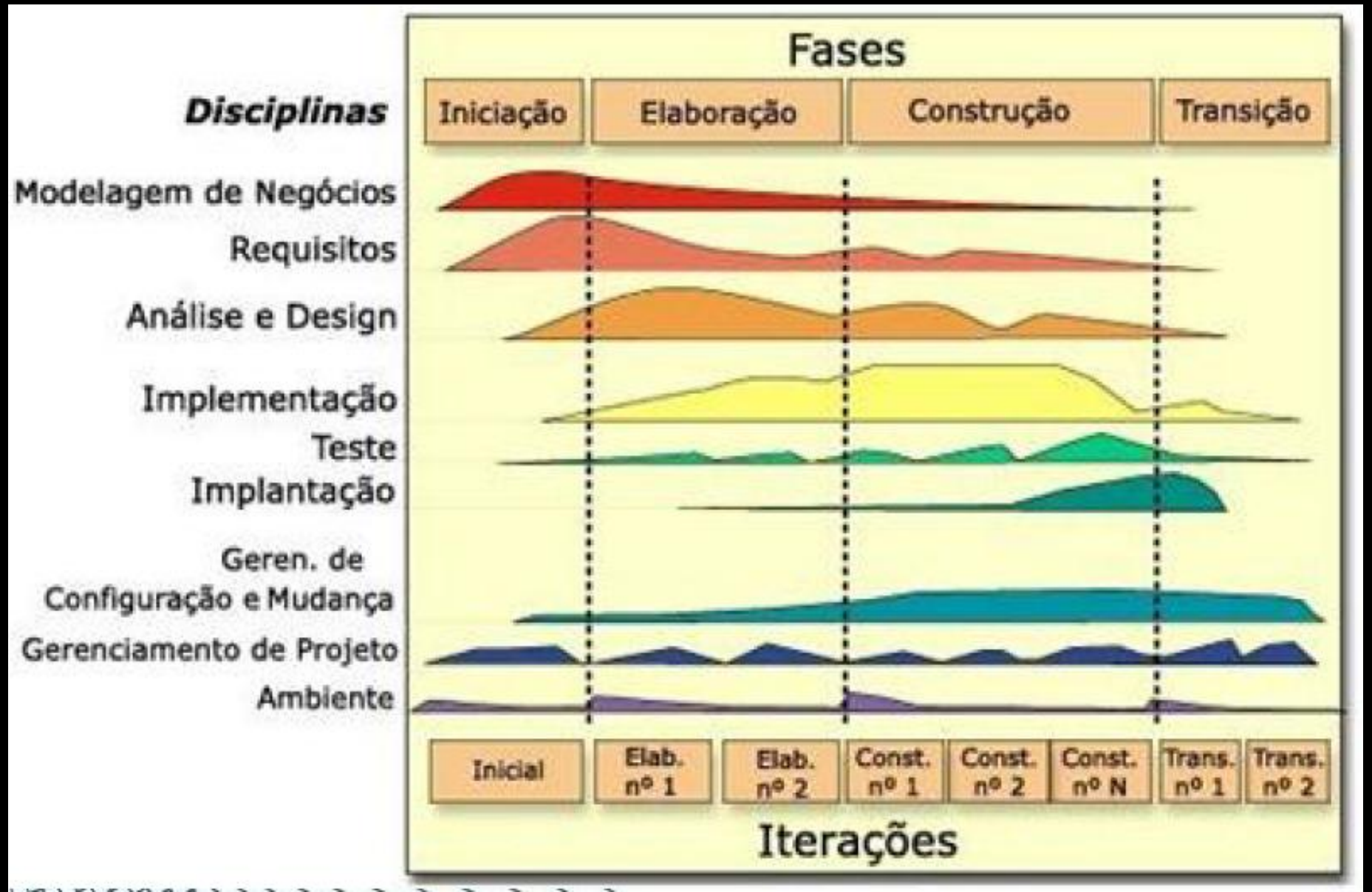
- Podemos dizer que o PU foi originado da Engenharia de Software baseada em componentes.
- O RUP é um modelo derivado do UML e do PU.
- Trata-se de um modelo híbrido.
  - Traz elementos de todos os modelos genéricos.
  - Apoia a iteração.
  - Ilustra boas práticas de especificação e projeto.

# Processos de Software – Processo Unificado (PU) – *Rational Unified Process (*RUP)

- Descrito a partir de três perspectivas:
  - Dinâmica – fases do modelo ao longo do tempo (4 fases).
  - Estática – atividades realizadas no processo (fluxo, workflow).
  - Prática – boas práticas a serem usadas durante o processo.

# Processos de Software – Processo Unificado (PU) – *Rational Unified Process (*RUP)

*Chamado gráfico das baleias



# Processos de Software – Processo Unificado (PU) – *Rational Unified Process (*RUP)

- Modelo constituído por 4 fases discretas:
  - **Inicialização/Concepção**: estabelecer um *business case* para o sistema. Avaliação do ambiente de negócio em relação à contribuição de um sistema para o negócio. (como era e como ficará)(escopo)
  - **Elaboração**: desenvolver um entendimento do domínio do problema, estabelecer um *framework*, desenvolver um plano de projeto e identificar os riscos principais do projeto. (modelo de requisitos para o sistema).(arquitetura).

# Processos de Software – Processo Unificado (PU) – *Rational Unified Process (*RUP)

- Modelo constituído por 4 fases discretas:
  - **Construção**: relacionada ao projeto, programação e teste de sistema. As partes são desenvolvidas separadamente e integradas. Software funcional + documentação. (desenvolvimento)
  - **Transição**: fase final do RUP. Transferência do desenvolvimento para o usuário. Entrada do sistema em produção. Fase onerosa e problemática. (implantação).

# Processos de Software– Processo Unificado – *Rational Unified Process* - RUP

- Cada fase pode ser realizada de forma **iterativa**, com resultados desenvolvidos **incrementalmente**.
- O conjunto total de fases pode ser realizado de forma incremental.
- Atacando primeiramente os pontos críticos e avaliando fortemente o risco e viabilidade.

# Processos de Software – Processo Unificado (PU) – *Rational Unified Process (*RUP)

- O RUP recomenda 6 melhores práticas fundamentais (linhas mestras):
  - Desenvolver o software iterativamente: incrementos de software priorizados e entregues.
  - Gerenciar requisitos: documentação e acompanhamento das mudanças dos requisitos.
  - Usar arquiteturas baseadas em componentes: estruturar a arquitetura de sistema de componentes. (o componente pode ser um incremento)
  - Modelar o software visualmente: modelos gráficos UML.
  - Verificar a qualidade do software: atendam aos padrões de qualidade da organização.
  - Controlar as mudanças do software: usando um SGM e procedimentos.

# Processos de Software – Processo Unificado (PU) – *Rational Unified Process (*RUP)

- Visão estática do RUP (os *Workflows*):
  - Fases são dinâmicas e tem objetivos.
  - *Workflows* são estáticos e constituem atividades técnicas que não estão associadas a uma única fase.
  - As fases não estão associadas aos *workflows* específicos.

# Processos de Software – Processo Unificado (PU) – *Rational Unified Process (*RUP)

- Visão estática do RUP (os *Workflows*):
- Principais
  - Modelagem de negócios: processos de negócio são modelados **usando casos de uso de negócios**.
  - Requisitos: agentes que interagem com o sistemas são identificados e os UCs são desenvolvidos para modelar os requisitos de sistema.
  - Análise e projeto: modelo de projeto é criado e documentado usando modelos de arquitetura, modelos de componente, modelos de objeto e modelos de sequencia.
  - Implementação: componentes de sistema são implementados e estruturados em subsistemas de implementação.

# Processos de Software – Processo Unificado (PU) – *Rational Unified Process (*RUP)

- Visão estática do RUP (os Workflows):
  - Teste: processo iterativo realizado em conjunto com a implementação.
  - Implantação: versão do produto é criada, distribuída aos usuários e instalada. (vide ferramentas case)
- Auxiliares
  - Gerenciamento de configuração e mudanças: gerencia as mudanças do sistema. (versões)
  - Gerenciamento de projetos: gerencia o desenvolvimento do sistema. (em que fase se encontra e como)
  - Ambiente: está relacionado à disponibilização de ferramentas apropriadas de software para a equipe de desenvolvimento.

# Exercício

- Crie uma mini história que para ser resolvida utilize o modelo de software PU/RUP também citando detalhes da solução e desenvolvimento.
  - **Tudo deve ser descrito com escrita descritiva focando em organização logica do texto e separando em fases como o PU/RUP trabalha.**
  - **Envie para o sistema SIGAA.**

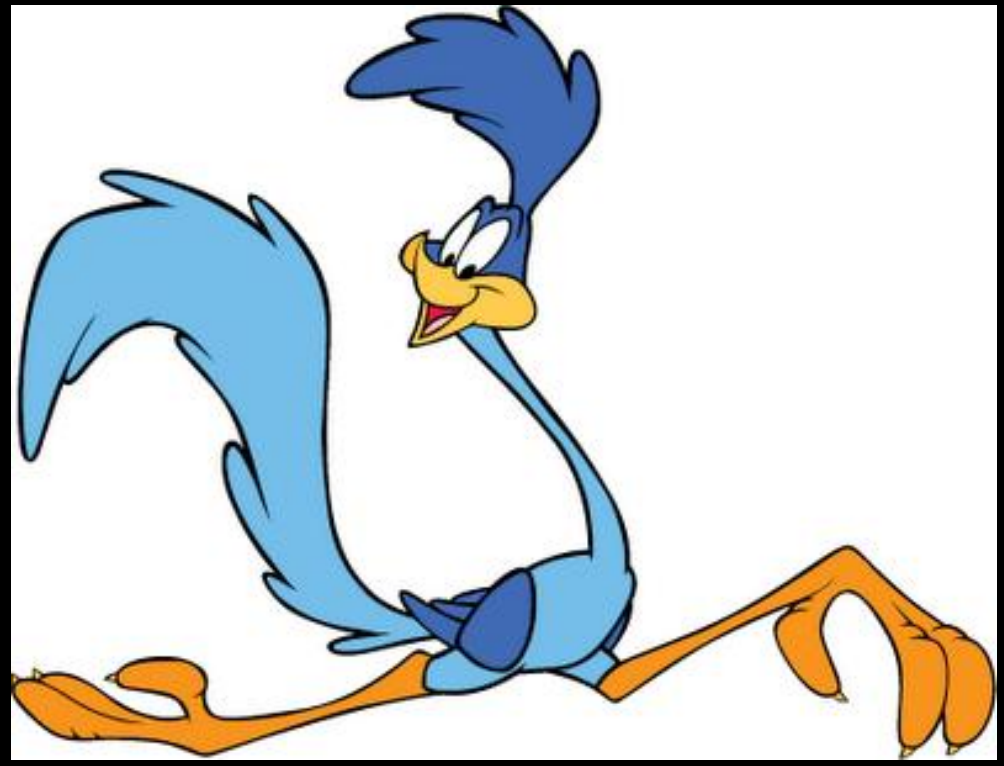
# Exercício – Ferramentas Case

- O que são e exemplifique as **ferramentas case**.
  - Modo mini seminário.
  - Pode ser em quatro “personas”.
  - **Não obrigatório.**

# Conteúdo Programático.

- **Modernos**
  - Engenharia de Software baseada em componentes (reaproveitamento).
  - Processo Unificado (voltado a grandes projetos).
  - História dos métodos ágeis;
  - Programação Extrema (XP) (desburocratização).
  - Scrum, Kanban (gerenciamento de software).

# Métodos Ágeis



# Processos de Software – Métodos Ágeis Manifesto

- Overhead (exagero) dominando o processo de software. (mudança constante durante o desenvolvimento)
- Inicio da década de 90 alguns desenvolvedores se manifestaram sobre a realidade encontrada no desenvolvimento de software de pequeno porte.
- Manifesto ágil e declaração de interdependência.
- 2001 no Resort Snowbird, em Utah para seu rascunho (www.agilemanifesto.org).

# Processos de Software – Métodos Ágeis Manifesto

- “Estamos descobrindo melhores maneiras de desenvolver software, fazendo-o e ajudando outros a fazê-lo. Ao longo deste trabalho, começamos a **valoriza**:
  - **Indivíduos e interações em vez de processos e ferramentas.**
  - **Software funcional em vez de documentação extensiva.**
  - **Colaboração com o cliente em vez de negociação de contrato.**
  - **Resposta à mudança em vez de seguimento de um plano.**
- Ou seja, mesmo havendo valor nos itens da direita, valorizamos mais os itens da esquerda.”

# Processos de Software – Métodos Ágeis Manifesto

- Autores
  - Kent Beck, Mike Beedel, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martim Fowler, James Grenning, Jim Highsmith, Andrew Hung, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, **Ken Scwaber, Jeff Sutherland, Dave Thomas**.

# Processos de Software – Métodos Ágeis Manifesto

- Doze princípios do manifesto ágil.
  - Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor.
  - Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas.
  - Entregar software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos.
  - Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto.

# Processos de Software – Métodos Ágeis Manifesto


- Doze princípios do manifesto ágil.
  - Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.
  - 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.
  - Software funcional é a medida primária de progresso.
  - Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes.

# Processos de Software – Métodos Ágeis Manifesto

- Doze princípios do manifesto ágil.
  - Contínua atenção à excelência técnica e bom design, aumenta a agilidade.
  - Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.
  - As melhores arquiteturas, requisitos e designs emergem de times auto organizáveis.
  - Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.

# Processos de Software – Métodos Ágeis Declaração de Independência - DOI

- Elaborada em 2005.
- Foco na Gestão de Projetos Ágeis.
- “Somos uma comunidade de líderes de projeto que tem sido altamente bem-sucedida em entregar resultados. Para alcança tais resultados:
  - Aumentamos o retorno do investimento, tornando o fluxo contínuo de valor o nosso foco.
  - Entregamos resultados confiáveis, engajando os clientes em interações frequentes e propriedade compartilhada.
  - Esperamos incertezas e gerenciamos levando-as em conta, por meio de iterações, antecipação e adaptação.

# Processos de Software – Métodos Ágeis Declaração de Independência - DOI

- Promovemos criatividade e inovação reconhecendo que os indivíduos são a fonte última de valor 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 maio de estratégias situacionais específicas, processos e práticas.”

# Processos de Software – Métodos Ágeis

- Negócio atual extremamente voltado a mudanças e surgimento de serviços concorrentes.
- Impossível então gerar requisito completos e estáveis de um software.
- Os processos tradicionais não estão voltados ao desenvolvimento de software.
- “Para sistemas de negócios específicos, processos de desenvolvimento voltados para a o desenvolvimento e entrega rápidos são essenciais.”

# Processos de Software – Métodos Ágeis
# Processos de desenvolvimento rápido

- Projetados para criar software útil rapidamente.
- Em geral são iterativos nos quais a especificação, o projeto, o desenvolvimento e o teste são intercalados.
- Processos de especificação, projeto e implementação são concorrentes.
- Documentação mínima (necessária) gerada pelo ambiente de programação.

# Processos de Software – Métodos Ágeis
# Processos de desenvolvimento rápido

- Documento de requisitos define somente as características mais importantes do sistema.
- Sistema desenvolvido em uma série e incrementos.
- Usuários finais e *Stakeholders* da especificação e avaliação dos incrementos, podem propor alterações e novos requisitos para um próximo incremento.
- Interfaces desenvolvidas utilizando um sistema de desenvolvimento interativo.

# Processos de Software – Métodos Ágeis
# Processos de desenvolvimento rápido

- **Vantagens:**
  - Entrega acelerada dos serviços ao cliente.
  - Engajamento do usuário com o sistema (envolvimento).
- **Problemas na implantação:**
  - Problemas de gerenciamento (mudanças rápidas, documentação, tecnologias não conhecidas).
  - Problemas de contrato (preço fixo?).
  - Problemas de validação (TDD).
  - Problemas de manutenção.

# Processos de Software – Métodos Ágeis
# Processos de desenvolvimento rápido

- **Prototipação evolucionária**: sinônimo de desenvolvimento de software incremental e neste caso o protótipo não é descartado, mas sim desenvolvido para atender aos requisitos do cliente.
  - Inicia pelos requisitos mais bem compreendidos e com a prioridade mais alta. (vagos e baixa prioridade fica pra depois).
  - *Throw-away*: começa com requisitos não muito bem compreendidos para saber mais sobre eles, requisitos simples talvez nunca sejam implementados.

# Processos de Software – Métodos Ágeis
# Processos de desenvolvimento rápido

- Seus implementadores enfatizam tanto sua utilização que as vezes esquecem de suas desvantagens:
  - Envolvimento com o cliente (cadê ele?).
  - Membros da equipe podem não interagir bem com outros membros da equipe.
  - Priorização de mudanças pode ser bem difícil, principalmente onde existem muitos *Stakeholders*.
  - Manter a simplicidade requer trabalho extra.

# Conteúdo Programático.

- **Modernos**
  - Engenharia de Software baseada em componentes (reaproveitamento).
  - Processo Unificado (voltado a grandes projetos).
  - História dos métodos ágeis;
  - Programação Extrema (XP) (desburocratização).
  - Scrum, Kanban (gerenciamento de software).

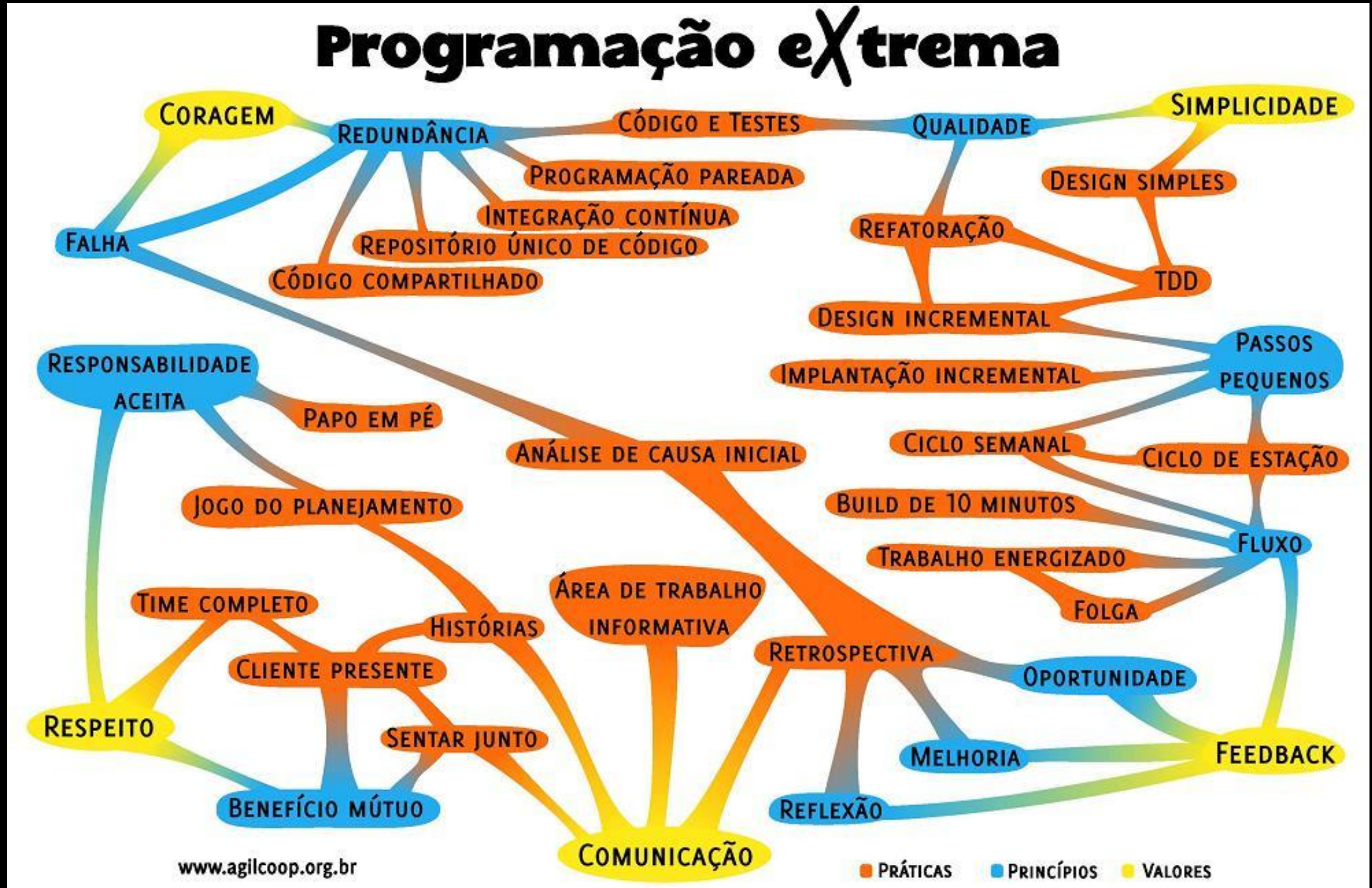
# Método Ágil - *Extreme Programming* (XP)

# Processos de Software – Métodos Ágeis *Extreme Programming* – Características

- Foi durante um bom tempo a ferramenta ágil mais conhecida do mercado.
- Tem seu nome do **desenvolvimento iterativo** e **envolvimento extremo do cliente**.
- Todos os seus requisitos são expressos como **cenários (histórias de usuários)** e são implementados diretamente como uma série de **tarefas**.
- Os programadores **trabalham em pares**.
- Testes são desenvolvidos antes da escrita do código.
- Todos os testes executados com sucesso quando um novo código é integrado ao sistema.

# Processos de Software – Métodos Ágeis *Extreme Programming* – Características

- Pequeno espaço de tempo entre as ***releases*** do sistema que apoia assim o desenvolvimento **incrementa**.
- Envolvimento do cliente apoiado pelo engajamento em tempo integral deste na equipe do de desenvolvimento sendo responsável pela definição de teste de aceitação do sistema. **(analogia do taxi)**
- Dentro da tríplice restrição ele está focado no escopo.


Programação eXtrema
Coragem
Redundância
Código e Testes
Qualidade
Simplicidade
Programação pareada
Design simples
Integração contínua
Refatoração
Falha
Repositório único de código
Código compartilhado
TDD
Design incremental
Passos pequenos
Responsabilidade aceita
Implantação incremental
Papo em pé
Análise de causa inicial
Ciclo semanal
Ciclo de estação
Jogo do planejamento
Build de 10 minutos
Fluxo
Trabalho energizado
Área de trabalho informativa
Time completo
Folga
Histórias
Retrospectiva
Cliente presente
Oportunidade
Respeito
Sentar junto
Melhoria
Feedback
Benefício mútuo
Reflexão
Comunicação
www.agilcoop.org.br
Práticas
Princípios
Valores

# Processos de Software – Métodos Ágeis
# *Extreme Programming* – Princípios

- *Feedback* rápido
- Presumir simplicidade
- Mudanças incrementais
- Abraçar mudanças
- Trabalho de alta qualidade.

# Processos de Software – Métodos Ágeis
# Extreme Programming – Valores

- Comunicação.
- Simplicidade.
- *Feedback.*
- Coragem.
- Respeito.

# Processos de Software – Métodos Ágeis
# *Extreme Programming* – Práticas

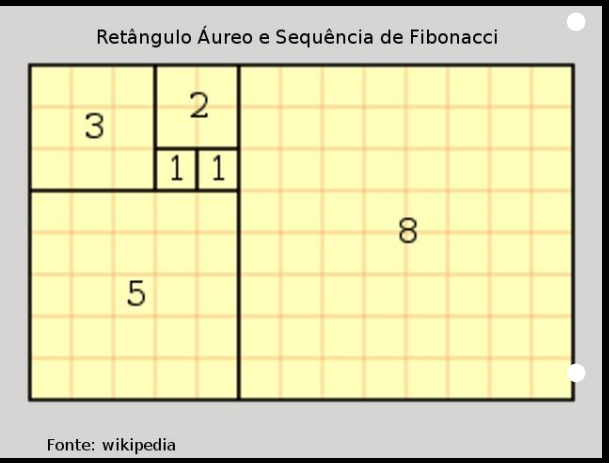
- Jogo de planejamento.
- **Planejamento incremental.**
- **Pequenas versões** (menores ainda que no RUP).
- Metáfora.
- **Projeto Simples** (não confundir com projeto fácil).
- Time coeso (pessoas engajadas e multidisciplinar).
- **Testes de aceitação/Cliente on-site.**
- **Ritmo sustentável** (trabalho motivado e hora extra quando trouxer vantagens).
- Reuniões em pé.

# Processos de Software – Métodos Ágeis
# Extreme Programming – Práticas

- **Posse coletiva** (o código não tem dono).
- **Programação em pares.**
- Padrões de codificação.
- **Desenvolvimento orientado a testes.** (TDD, *Testfirst*)
- **Refatoração.**
- **Integração contínua.**

# Processos de Software – Métodos Ágeis
# *Extreme Programming* – Cartão de História

- **Template de Cartão de Historia**
  - *Como um [usuário papel], quero [meta], para que eu possa [motivo].*




- Após a sua definição, a equipe de desenvolvimento deverá dividi-lo em tarefas e estimar o esforço e os recursos necessários para implementação.
  - **Plainning poker.**
- O cliente prioriza as histórias para implementação.
- Mudança sem histórias não implementadas podem gerar alterações ou descarte.

# Processos de Software – Métodos Ágeis *Extreme Programming* – Abordagem Extrema

- Novas versões podem se compiladas várias vezes por dia.
- Incrementos são entregues aproximadamente a cada duas semanas.
- Quando o desenvolvedor compila o sistema para criar uma nova versão, ele deve executar todos os testes automatizados existentes (**Ferramentas Case**) e os testes para a nova funcionalidade.

# Processos de Software – Métodos Ágeis
# *Extreme Programming* – Testes

- O processo de testes dentro do XP é mais enfatizado que em outros métodos ágeis.
- Características:
  - TDD – Test Driven Development (test-firt): define tanto uma interface quanto um comportamento para a funcionalidade. (entrada, testes, saída).
  - Desenvolvimento incremental de teste a partir de **cenários**.
  - Envolvimento do usuário no desenvolvimento e validação de testes.
  - O uso de ferramentas de teste **automatizadas**: teste escrito como um componente executável antes do código. Deve possibilitar a entrada e verificar se a saída atende. **(maquinas de teste) (Ferramentas Case)**

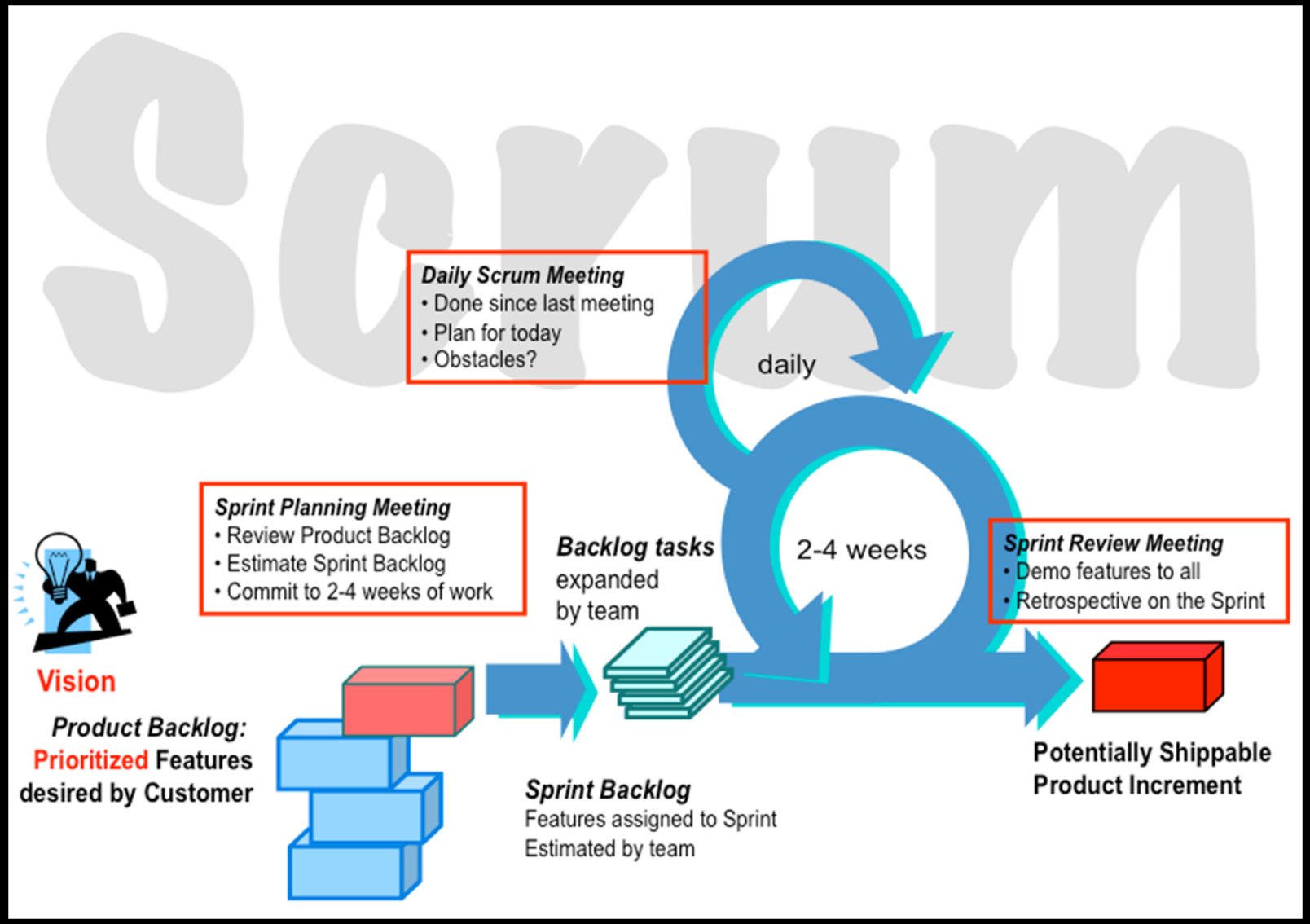
# Conteúdo Programático.

- **Modernos**
  - Engenharia de Software baseada em componentes (reaproveitamento).
  - Processo Unificado (voltado a grandes projetos).
  - História dos métodos ágeis;
  - Programação Extrema (XP) (desburocratização).
  - Scrum, Kanban (gerenciamento de software).

# Scrum e kamban

- Scrum e Kamban são alternativas de gerenciamento de projetos muito usadas, inclusive entre administradores de empresa.
- Ambas tem suas nomenclatura como cards ou tarefas... mas ambas focam em destacar as atividades dentro de fases de um processo temporal com figurações como post-tis.
- Segue alguns links de APPs:
  - Easy Kamban - https://play.google.com/store/apps/details?id=br.com.umdesenvolvedor.easy_kanban
  - Scrum Management - https://play.google.com/store/apps/details?id=com.company.scrum.management

# Método Ágil - Scrum



# Processos de Software – Métodos Ágeis Scrum

- O Scrum é um pacote com mais de 50 modelos de relatórios completos, mais de 9 livros que indicam as melhores práticas no gerenciamento de projetos, algumas framework em software com ferramenta CASE para auxiliar os seus implementadores e uma ferramenta de Gerenciamento de Projetos integrada ao pacote de escritório da sua organização?
  - Não.

# Processos de Software – Métodos Ágeis Scrum

- É um processo iterativo e incremental para ao desenvolvimento de produtos e gerenciamento de projetos.
- **Está mais para framework** do que para uma metodologia.
- Ele não te dirá o que fazer, apenas te dará transparência para enfrentar os desafios do dia a dia, a decisão é tua.

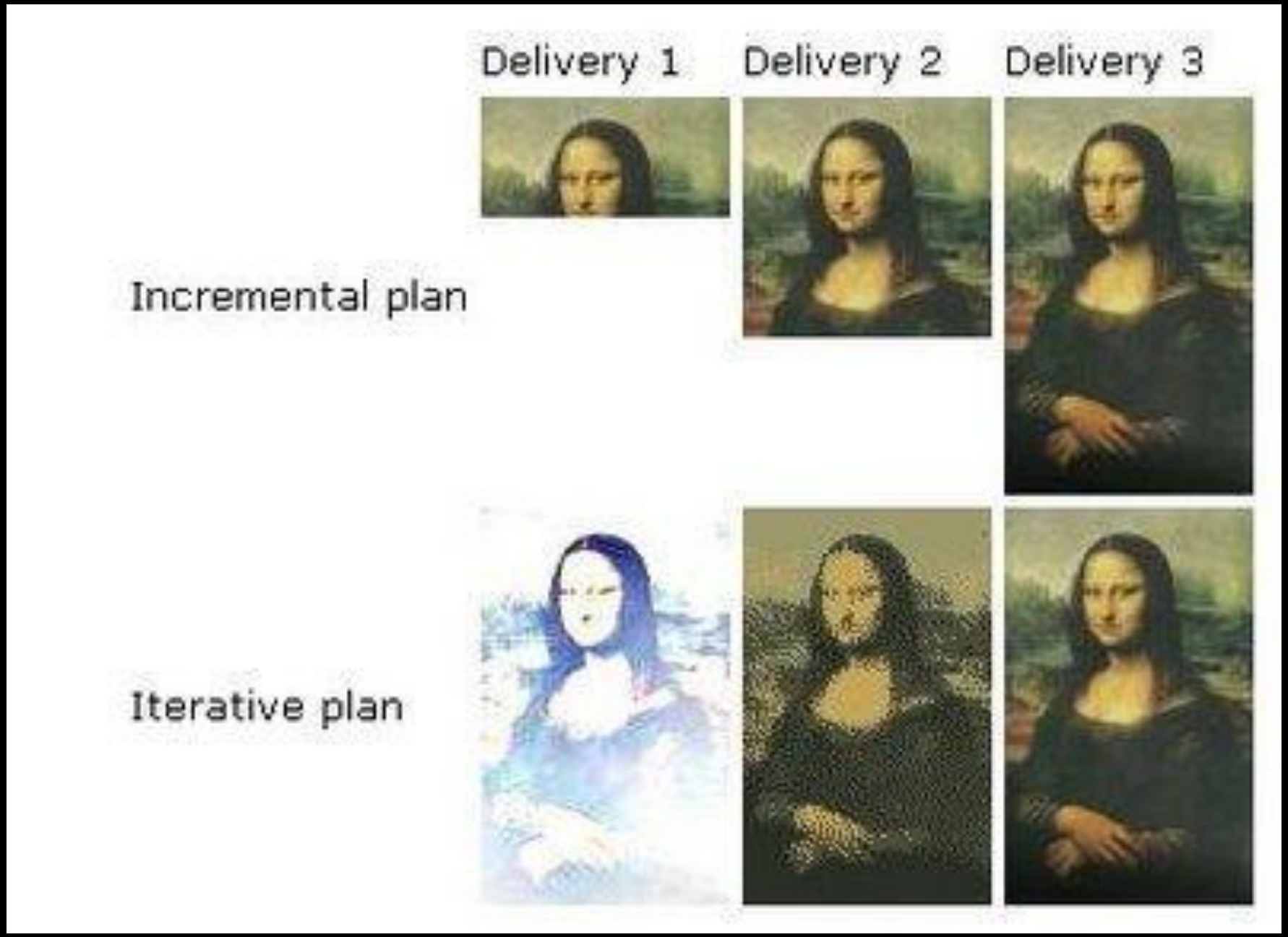
# Processos de Software – Métodos Ágeis Scrum

- Método ágil para gerenciamento de projetos baseado em **times pequenos** e auto organizados.
- Apresenta forte-visibilidade e rápida adaptação a mudanças.
- Tomou nome quando Jeff Shutherland e Ken Shwaber utilizaram seu corpo de conhecimento na Easel em 1994.

# Processos de Software – Métodos Ágeis Scrum

- Foi apresentado formalmente em 1996 na *Object-Oriented Programming, Systems, Languages, and Applications*.
- **RUP** é centrado em arquitetura e negócio o **Scrum** é centrado em valores, entregas reais de produtos.
- **Iterativo** para garantir inspeção e adaptação do produto e **incremental** para garanti-la no processo.

# Processos de Software – Incremental  X Iterativo



# Processos de Software – Métodos Ágeis Scrum

- “Estamos descobrindo melhores maneiras de desenvolver software, fazendo-o e ajudando outros a fazê-lo. Ao longo deste trabalho, começamos a valoriza:
  - Indivíduos e interações em vez de processos e ferramentas.
  - Software funcional em vez de documentação extensiva.
  - Colaboração com o cliente em vez de negociação de contrato.
  - Resposta à mudança em vez de seguimento de um plano.
- Ou seja, mesmo havendo valor nos itens da direita, valorizamos mais os itens da esquerda.”

# Processos de Software – Métodos Ágeis
# Scrum – 3 pilares

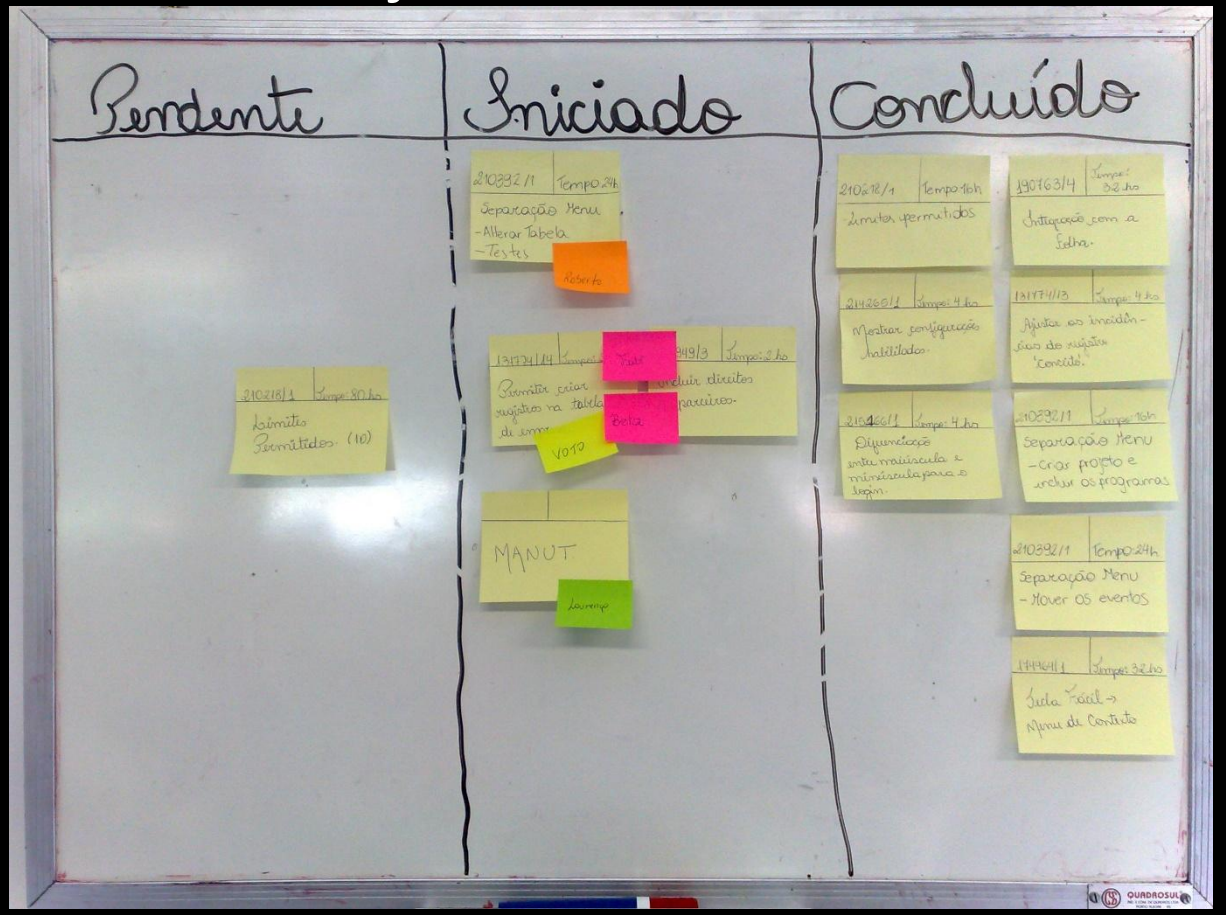
- Nenhuma nova prática quando integrada ao Scrum pode derrubar um destes pilares, senão já não é mais Scrum.
- **Transparência**: os aspectos do processo que afetam o resultado devem ser visíveis para aqueles que gerenciam os resultados. Tudo que se vê deve ser conhecido. Se alguém acha que algo está feito, isso quer dizer que está "pronto".

# Processos de Software – Métodos Ágeis
# Scrum – 3 pilares

- **Inspeção**: detecção de variâncias inaceitáveis no processo devem ser inspecionadas e detectadas.
  - **Daily Scrum** (reunião diária): progresso em direção ao objetivo do Sprint e adaptações para o próximo dia de trabalho.
  - **Sprint Review** e **Sprint Planning**: inspecionar o progresso em direção à meta da Sprint e adaptações que otimizem o valor do próximo Sprint.
  - **Sprint Retrospective**: revisar o Sprint que passou e determinar quais adaptações devem ser feitas para o próximo.

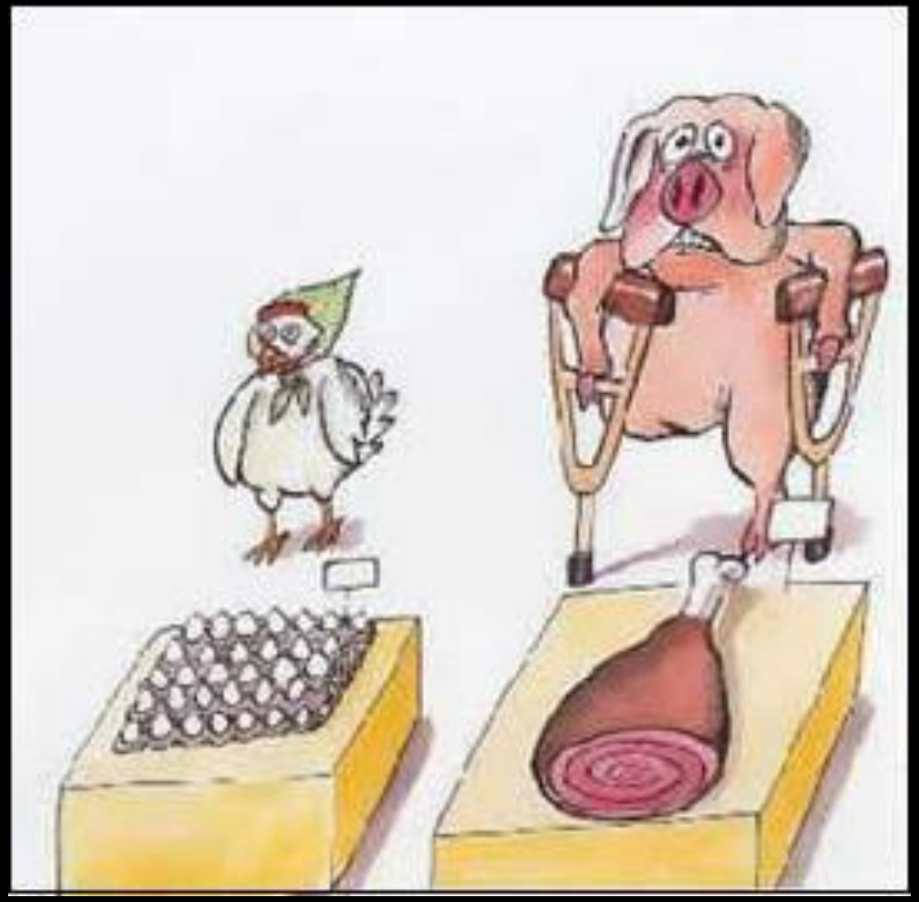
# Processos de Software – Métodos Ágeis
# Scrum – 3 pilares

- **Adaptação**: se for detectado no decorrer da inspeção que um ou mais aspectos do processo saiu dos trilhos e que o produto final será inaceitável, eles devem ser ajustados.



# Processos de Software – Métodos Ágeis
# Scrum – Papéis no Scrum

- **Comprometidos**: focados no projeto e com compromisso direto.
- **Envolvidos**: colaboram apenas.



# Processos de Software – Métodos Ágeis
# Scrum – Papéis no Scrum

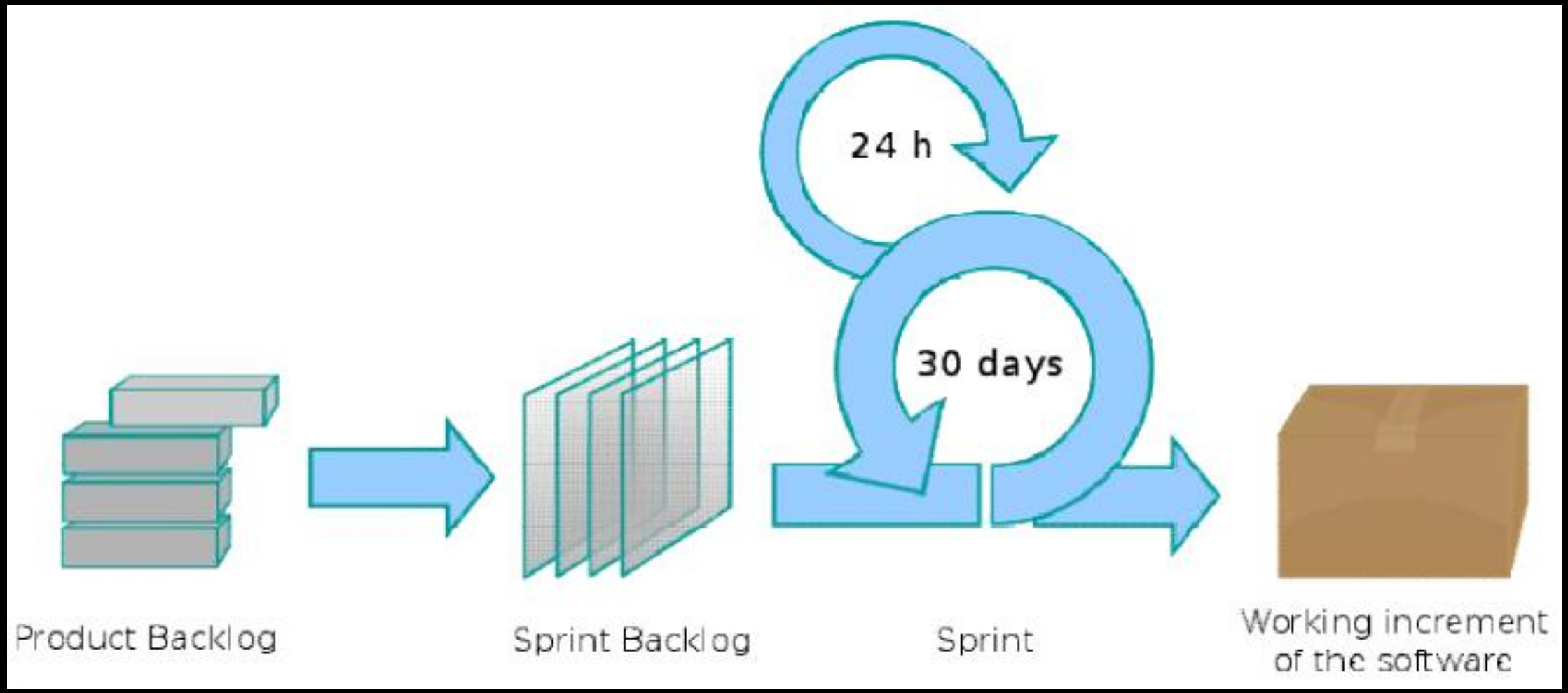
- ***Product Owner***:
  - Faz o Macro Management.
  - Tem total decisão sobre o produto.
  - Responsável por garantir o ROI.
  - Responsável por garantir as necessidades do cliente.
  - Proxy de comunicação em ambientes com mais de um cliente.
  - **Precisa ter disponibilidade e não estar dentro do time.**

# Processos de Software – Métodos Ágeis
# Scrum – Papéis no Scrum

- **Scrum Master**:
  - Responsável por remover os impedimentos do time.
  - Garante o uso do Scrum.
  - Protege o time de interferências externas.
  - Não deve ser PO (*Product Owner*) nem fazer parte do time.
- **Time**:
  - Multidisciplinar.
  - Auto organizado.
  - Produz o produto.
  - Metas compartilhadas.
  - É quem faz o micro management.

# Processos de Software – Métodos Ágeis
# Scrum – Fluxo do Scrum

- Composto por papéis, reuniões e artefatos.
- **Artefatos**: *Product Backlog, Sprint Backlog* e **Burndown. Chart**.



# Processos de Software – Métodos Ágeis
# Scrum – Fluxo do Scrum

- O *Produt Owner* define a visão do produto.
- Para tanto ele recolhe informações junto ao cliente, usuários finais, time, gerentes, stk, etc.
- A visão define o que o *Produt Owner* espera que seja entregue como produto final do projeto e sabe-se que ele foi finalizado quando a visão foi atendida.
- Definição de Pronto.
- Escopo aberto.

# Processos de Software – Métodos Ágeis Scrum – Fluxo do Scrum – *Product Backlog*

- O PO cria uma lista inicial de necessidades (*Product Backlog*) com o auxílio do Scrum Master.
- O *Product Backlog* contem os requisitos para o produto que a equipe está desenvolvendo.
- O PO é o responsável pelo *Product Backlog*: conteúdo, disponibilidade e priorização.
- O *Product Backlog* é o **documento dinâmico**.
- Não existe um padrão para documentação técnica, pode ser UC, *Use Stories*.
- Um item do *backlog* deverá sempre caber em uma *sprint*.

# Processos de Software – Métodos Ágeis Scrum – Fluxo do Scrum – *Product Backlog*

- Quanto mais no topo do *Product Backlog*, mais prioritária será a necessidade e automaticamente mais detalhada ela estará.
- Quanto mais no fundo, menos prioritária será a necessidade e automaticamente menos detalhada ela estará.

# Processos de Software – Métodos Ágeis Scrum – Fluxo do Scrum – *Product Backlog*

- **Template de Cartão de Historia**
- *Como um [usuário papel], quero [meta], para que eu possa [motivo].*
- Após a sua definição, a equipe de desenvolvimento deverá dividi-lo em **tarefas** e estimar o esforço e os recursos necessários para implementação.
  - ***Plainning poker***.

# Processos de Software – Métodos Ágeis
# Scrum – Fluxo do Scrum – *Sprint Planning Meeting*

- Planejamento da iteração.
- 8 horas para um Sprint de 1 mês.
- 5% do tamanho total do Sprint para essa reunião.
- No seu final deveremos ter o Sprint Backlog.
- Divide-se em duas partes.
  - Primeira parte (metade) decide-se o que vai fazer no Sprint.
  - Segunda parte (metade) o time decide como vai construir as funcionalidades em um incremento do produto durante a Sprint.

# Processos de Software – Métodos Ágeis
# Scrum – Fluxo do Scrum – *Sprint Planning Meeting*

- **Primeira parte:**
  - Tratamento do “o que?”.
  - O PO apresenta os itens de maior prioridade do *Product Backlog*.
  - Trabalho em conjunto para descobrir qual funcionalidade deverá ser desenvolvida no próximo *Sprint*.
  - Tem como entrada o *Product Backlog*, o último incremento do produto, a capacidade do Time e a performance do time no *Sprint* anterior.
  - A decisão dos itens que entrarão no *Sprint* cabe ao time.
  - Define-se a meta da *Sprint*.
  - SM como facilitador.

# Processos de Software – Métodos Ágeis
# Scrum – Fluxo do Scrum – *Sprint Planning Meeting*

- **Segunda parte:**
  - Tratamento do “como?”
  - Definir como o que foi definido vai ser um incremento pronto e implementado.
  - Enquanto projeta o como o time decompõe o trabalho em uma lista de tarefas a serem realizadas em menos de um dia (*Sprint Backlog*).
  - O time se auto organiza para delegar e se encarregar do trabalho.
  - O PO está presente para esclarecer dúvidas e efetuar mudanças.

# Processos de Software – Métodos Ágeis
# Scrum – Fluxo do Scrum – *Sprint Planning Meeting*

- **Segunda parte:**
  - Caso o time detecte que tem muito trabalho, isto poderá ser negociado com o PO.
  - Podem ser convidadas outras pessoas para participar da reunião.
  - O time acaba verificando que deve confiar em si próprio e isto gera a auto-organização.

# Processos de Software – Métodos Ágeis Scrum – Fluxo do Scrum – *Sprint Bakclog*

- Deverá estar pronto no final da *Sprint Plannning Meeting*.
- Caso se consiga 70% das tarefas necessárias para cumprimento do meta da Sprint já estará de bom tamanho.
- Deverá conter: itens de *backlog* selecionados, suas respectivas tarefas e a meta da *Sprint*.
- Diz respeito às atividades micro.

# Processos de Software – Métodos Ágeis
# Scrum – Fluxo do Scrum – *Sprint Bakclog*

# Processos de Software – Métodos Ágeis Scrum – Fluxo do Scrum – *Sprint Burndwn*

- É um gráfico do time, não do PO nem do SM.
- Tem como objetivo ajudar o time no *micro management*.
- O gráfico só evolui quando o que se tem está efetivamente pronto.
- Auxilia na mediação da velocidade da equipe por *sprint*.

# Processos de Software – Métodos Ágeis Scrum – Fluxo do Scrum – *Sprint*

- Seu tempo de duração inclui o *Sprint Planning* e o *Sprint Review* também.
- O SM facilita o trabalho do time removendo os impedimentos encontrados e garantindo a boa aplicação do Scrum.
- O time poderá consultar especialistas de domínio ou o PO.

# Processos de Software – Métodos Ágeis Scrum – Fluxo do Scrum – *Daily Meeting*

- Duração de 15 minutos:
  - O que fiz desde a última reunião?
  - O que pretendo fazer até a próxima?
  - Tive (estou tendo) algum impedimento?
- Ganho de visibilidade de como está o caminho para a meta e planeja o dia seguinte de trabalho.
- SM como facilitador.
- Esta reunião é para o time e não para o SM.
- Reunião de micro management.

# Processos de Software – Métodos Ágeis Scrum – Fluxo do Scrum – *Daily Meeting*

- Deverá ser analisada a inclusão de novas tarefas desde que elas estejam de acordo com o cumprimento da meta da *Sprint* e também não se muda a *baseline* de escopo da *Sprint* por conta dela. (azul no *Sprint Backlog*).
- Insere-se também os impedimento (vermelho no *Sprint Backlog*).
- Considera- se o *sprint* bem sucedido quando sua meta é alcançada.

# Processos de Software – Métodos Ágeis Scrum – Fluxo do Scrum – *Sprint Review*

- Realizada ao final do *Sprint*.
- Reunião de 4 horas para *Sprints* de um mês.
- Inclui pelo menos os seguintes itens:
  - O PO identifica o que foi e o que não foi feito.
  - O time discute o que correu bem e os problemas encontrados e como foram resolvidos.
  - O time demonstra o trabalho que foi feito e responde a questionamentos.
  - O PO discute o *Product Backlog* atual.
  - Todos discutem melhorias que poderão ser implementadas de acordo com o ocorrido.
  - Trata da melhoria do produto.

# Processos de Software – Métodos Ágeis Scrum – Fluxo do Scrum – Retrospectiva

- Tem foco na Inspeção-adaptação.
- Tem duração de 3 horas.
- O time é encorajado pelo SM para revisar o processo de trabalho de acordo com as práticas do Scrum.
- Sua finalidade é inspecionar como correu o último *Sprint* em se tratando de pessoas, relações entre elas, processos e ferramentas.
- Considera o que correu bem e o que poderia ter sido feito para ter sido melhor ainda.
- Identificação de melhorias para a próxima *Sprint* ao seu final.
- Melhoria de processo.

# Processos de Software – Métodos Ágeis Scrum – *Stories*, Temas e Epics

- *Epics* são grandes *user stories*, que ainda estão em formato bruto.
- Temas são pacotes de *user stories* relacionadas por algum grande objetivo de negócio.
- Analogamente o Tema é o mesmo que *package* de um caso de uso.
- Uma release pode ter 1 ou mais temas.
- No *planning poker* passou de 40 é um *Epic*.

# Processos de Software – Métodos Ágeis
# Scrum – *Releases X Sprints*

- ***Sprint*** deve entregar um incremento potencialmente entregável do produto com alta qualidade, testado, completo e pronto.
- ***Release*** é uma entrega que entrará em produção, de acordo com uma condição de satisfação real, clara e prioritária. Somente após a finalização de um *Release* é que um novo deverá ser planejado.

# Processos de Software – Métodos Ágeis
# Scrum – Mudanças durante a *Sprint*

- Deverá ser entregue no final da Sprint o que foi acordado com o PO:
- Caso ocorra mudanças:
  - As mudanças ocorrem desde que não atrapalhem a meta da Sprint.
  - O *time-box* da *sprint* é fixo, não muda devido ao fato das mudanças nem do conteúdo ou se insere horas-extras.
  - Pode haver mudanças desde que não impacte no timer da *sprint*.

# Processos de Software – Métodos Ágeis
# Scrum – Finalização da Sprint Antes da Hora

- Pode terminar antes da hora quando:
  - O time sente que não conseguirá atingir a meta.
  - O PO percebe que mudanças em fatores externos influenciarão diretamente no valor da meta da Sprint.
- Quando uma *Sprint* é finalizada, deve-se iniciar imediatamente o planejamento da próxima *Sprint*.

# Processos de Software – Métodos Ágeis
# Scrum – Scrum *of* Scrum

- Não existe um SM ou PO do *Scrum of Scrums*, trata-se de um conjunto de equipes.
- A meta deve ser compartilhada entre os times.
- Após reuniões diárias do time interno, realiza-se a diária com os representantes das outras equipes.

# Método Ágil - Kanban

# Processos de Software – Métodos Ágeis Kanban

- Significa placa visível.
- Originado como ferramenta para atendimento ao ***Lean***.
- Lean é um modelo da **Toyota** de produção chamado de produção enxuta ou *Lean* Manufacturing, surgiu no Japão, na fábrica de automóveis logo após a Segunda Guerra Mundial (1939-1945).
  - Aumento da eficiência da produção pela eliminação de desperdícios.
- Nasceu juntamente com o JIT, Jidoka, Kaizen como ferramentas para permitir a realização do *Lean*.

# Processos de Software – Métodos Ágeis Kanban

- O Kanban traz como grande inovação o conceito de **eliminar estoque** (estoque zero), os materiais e componentes agregados ao produto chegam ao momento exato da sua produção (JIT).
- Kanban **é um framework** Lean aplicável do desenvolvimento e operação.
- **Cartão de sinalização**, é um sinal visual produzido indicando que novo trabalho pode ser iniciado e que a atividade atual não coincide com o limite acordado. Isso não soa muito revolucionário nem parece afetar profundamente o desempenho, cultura, capacidade e maturidade de uma equipe e a organização na qual está inserida.

# Processos de Software – Métodos Ágeis Kanban

- Coloca-se um **Kanban** em peças ou partes específicas de uma linha de produção, para indicar a entrega de uma determinada quantidade. Quando se esgotarem todas as peças, o mesmo aviso é levado ao seu ponto de partida, onde se converte num novo pedido para mais peças. Quando for recebido o cartão ou quando não há nenhuma peça na caixa ou no local definido, então deve-se movimentar, produzir ou solicitar a produção da peça.
- O Kanban permite agilizar a entrega e a produção de peças.

# Processos de Software – Métodos Ágeis Kanban

- O Kanban é uma abordagem para **mudança gerencial**.
- Não é um processo ou ciclo de vida de gerenciamento de projetos ou de desenvolvimento de software.
- Inicia-se com o que estiver fazendo agora a partir daí se rastreia as atividades pelo sistema para iniciá-las quando o sinal de Kanban aparecer. (WIP – *work in progress*)

# Processos de Software – Métodos Ágeis Kanban

- O Kanban **expõe** gargalos, filas, variabilidade e desperdício.
- Incentiva a discussão sobre melhorias.
- Por utilizar *sistem pull* encoraja o comprometimento tardio.
- ***Pull system***, ou sistema puxado: sistema em que os times é que puxam o trabalho, na medida de sua capacidade, e não o oposto, quando o trabalho é **empurrado** à equipe – sistema de trabalho mais comum.

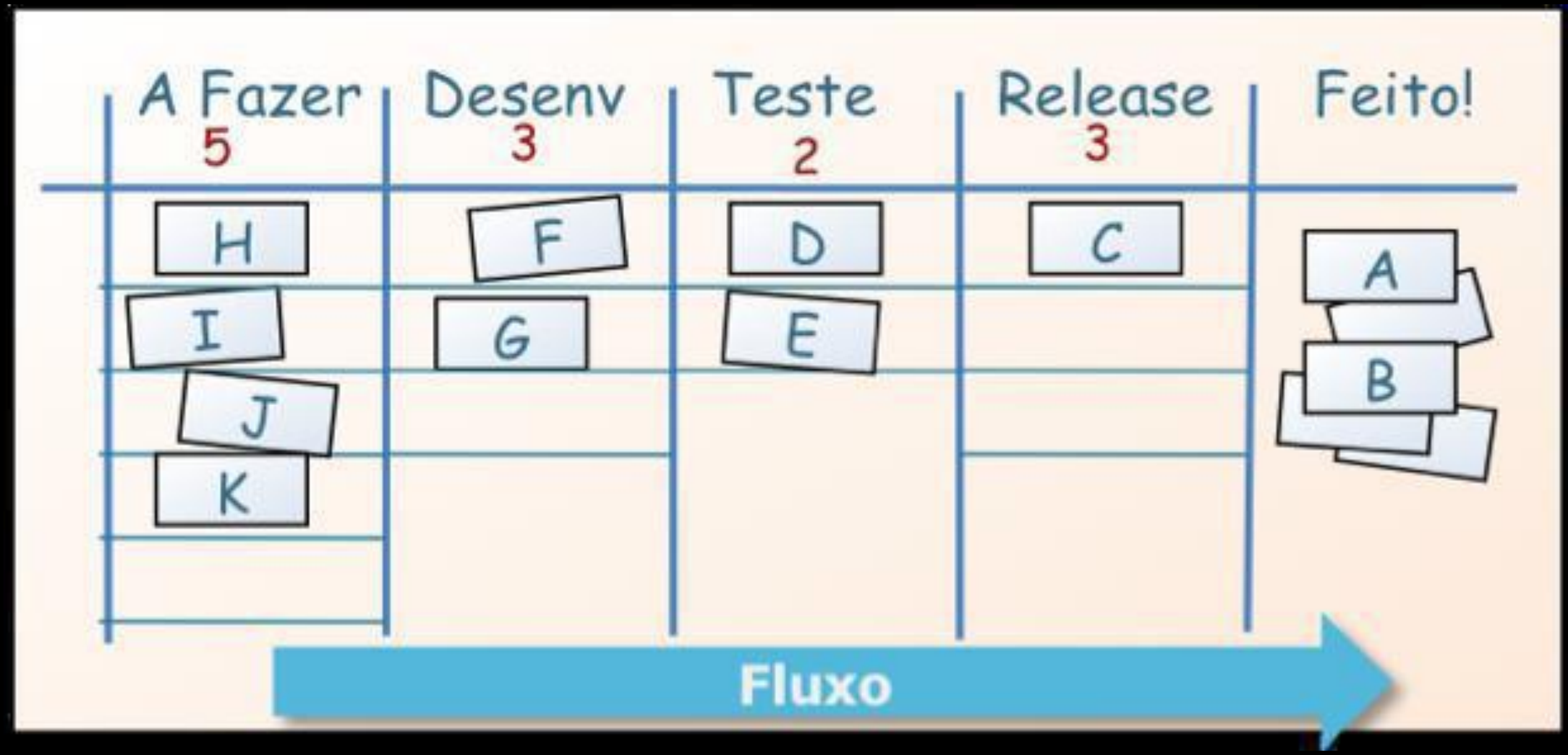
# Processos de Software – Métodos Ágeis
# Kanban de produção

- Kanban de Produção é o sinal que autoriza a produção de determinada quantidade de um item.
- Os **cartões** circulam entre o processo fornecedor e a área de fornecimento, sendo afixados junto às peças imediatamente após a produção e retirados após o consumo pelo cliente, retornando ao processo para autorizar a produção e reposição dos itens consumidos.

# Processos de Software – Métodos Ágeis Kanban de Movimentação

- Kanban de Movimentação, também chamado de Kanban de Transporte, é o sinal que autoriza a movimentação física de peças entre a área de fornecimento do processo fornecedor e a área de fornecimento do processo cliente.
- Os cartões são afixados nos produtos e levados a outro processo ou local, sendo retirados após o consumo e estando liberados para realizar novas compras na área de fornecimento do processo fornecedor.

# Processos de Software – Métodos Ágeis
# Kanban em Poucas Palavras

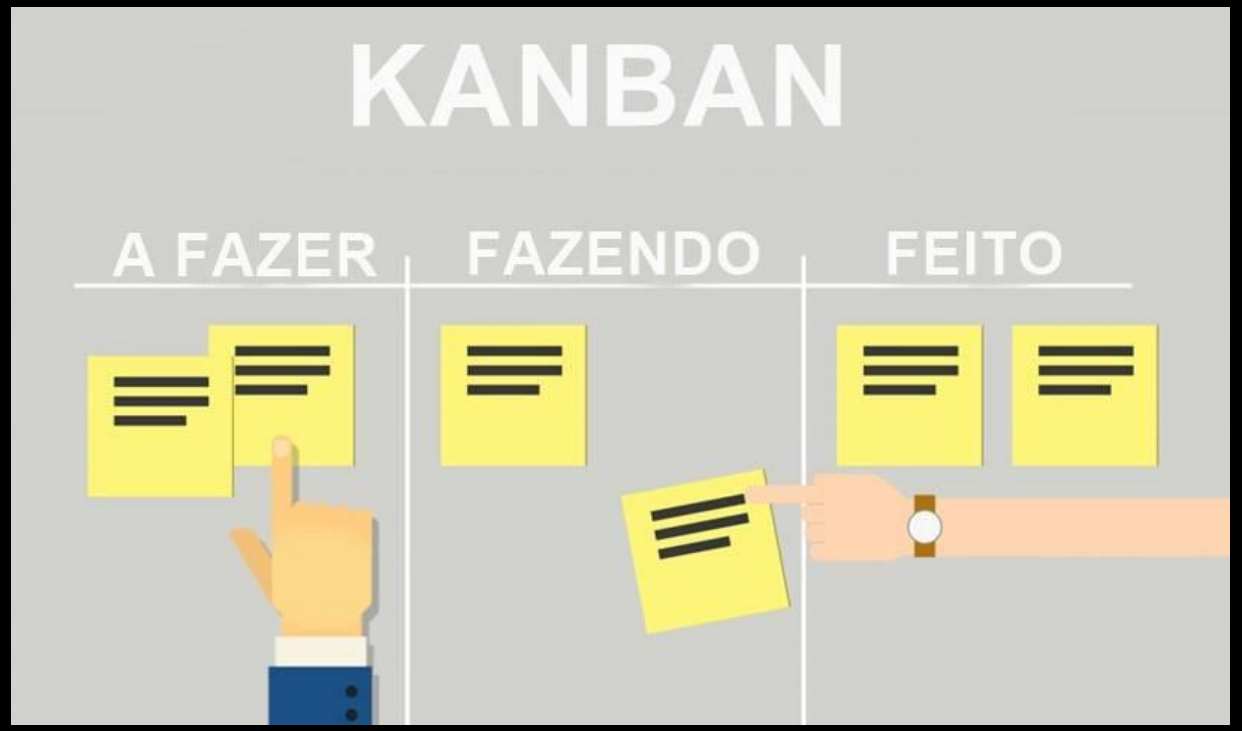


# Processos de Software – Métodos Ágeis Kanban em Poucas Palavras

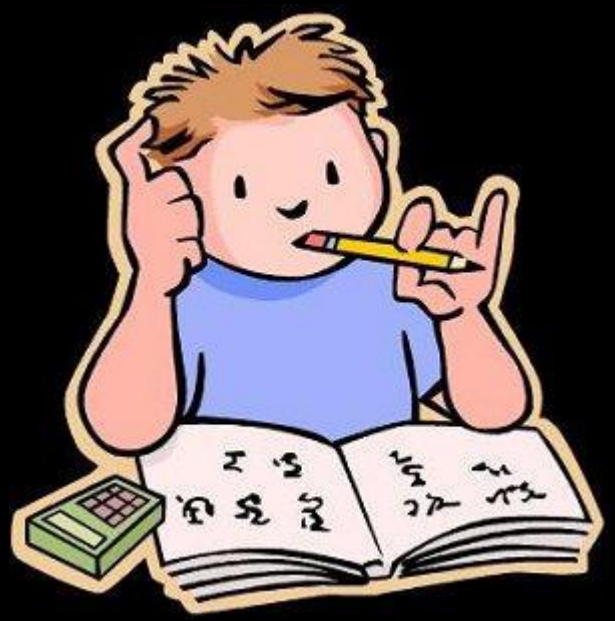
- Visualize o fluxo de trabalho.
  - **Divida o trabalho em partes**, escreva cada item em um cartão e coloque na parede.
  - **Use colunas nomeadas** para ilustrar onde cada item está no fluxo de trabalho.
- **Limite o trabalho em progresso** (WIP - *work in progress*) – associe limites explícitos para quantos itens podem estar em progresso em cada estado do fluxo de trabalho.

# Processos de Software – Métodos Ágeis Kanban em Poucas Palavras

- Acompanhe o tempo de execução da tarefa (tempo médio para completar um item, algumas vezes chamado de “tempo de ciclo”), otimize o processo para tornar o tempo de execução o **menor e mais previsível possível**.



# Exercícios



# Exercícios

- (BAZA – TI - 2012 – CESPE) Com relação a desenvolvimento de aplicativos web, Oracle e ferramentas CASE, julgue os itens subsequentes.
  - 1. Ferramentas de documentação, como geradores automáticos de relatórios, fazem parte do conjunto de ferramentas CASE empregadas no desenvolvimento de aplicativos.
  - 2. Ferramentas CASE são de uso específico no desenvolvimento de aplicativos web usando banco de dados.
  - 3. Metodologias de desenvolvimento XP contam com o desenvolvimento orientado a testes, que engloba duas etapas: escrever um teste automatizado e desenvolver um código adequado o suficiente para ter sucesso nesse teste.

# Exercícios

- 4. (TRE - ES - 2011 – CESPE) A metodologia *Rational Unified Process* (RUP) promove o envolvimento do cliente, bem como iterações e testes contínuos, o que torna o processo dependente de outros, apesar de reduzir os seus riscos. Já a metodologia *Extreme Programming* (XP) proporciona flexibilidade e agilidade, visto que, por meio dela, realiza-se a divisão de tarefas de forma específica.
- 5. (TRE – EBC - 2011 – CESPE) O XP segue um conjunto de valores, princípios e regras básicas que visam alcançar eficiência e efetividade no processo de desenvolvimento de software. Os valores são cinco: comunicação, simplicidade, feedback, coragem e respeito.

# Exercícios

- 6. (UFBA - 2009) No processo de *software* baseado em componentes, cada componente projetado para **reuso** é uma entidade executável independente, que deve manipular exceções.
- 7. (TRE - BA – 2010 - CESPE) Na engenharia de *software* baseada em componentes, na qual se supõe que partes do sistema já existam, o processo de desenvolvimento concentra-se mais na integração dessas partes que no seu desenvolvimento a partir do início. Essa abordagem é baseada em reuso para o desenvolvimento de sistemas de *software.*

# Gabarito

- 1. V
- 2. F
- 3. V
- 4. F
- 5. V
- 6. F
- 7. V

Livros Indicados e Utilizados: Pressmam e Sommerville.
Principal Material Utilizado para os Slides: Curso do Site
www.euvoupassar.com.br