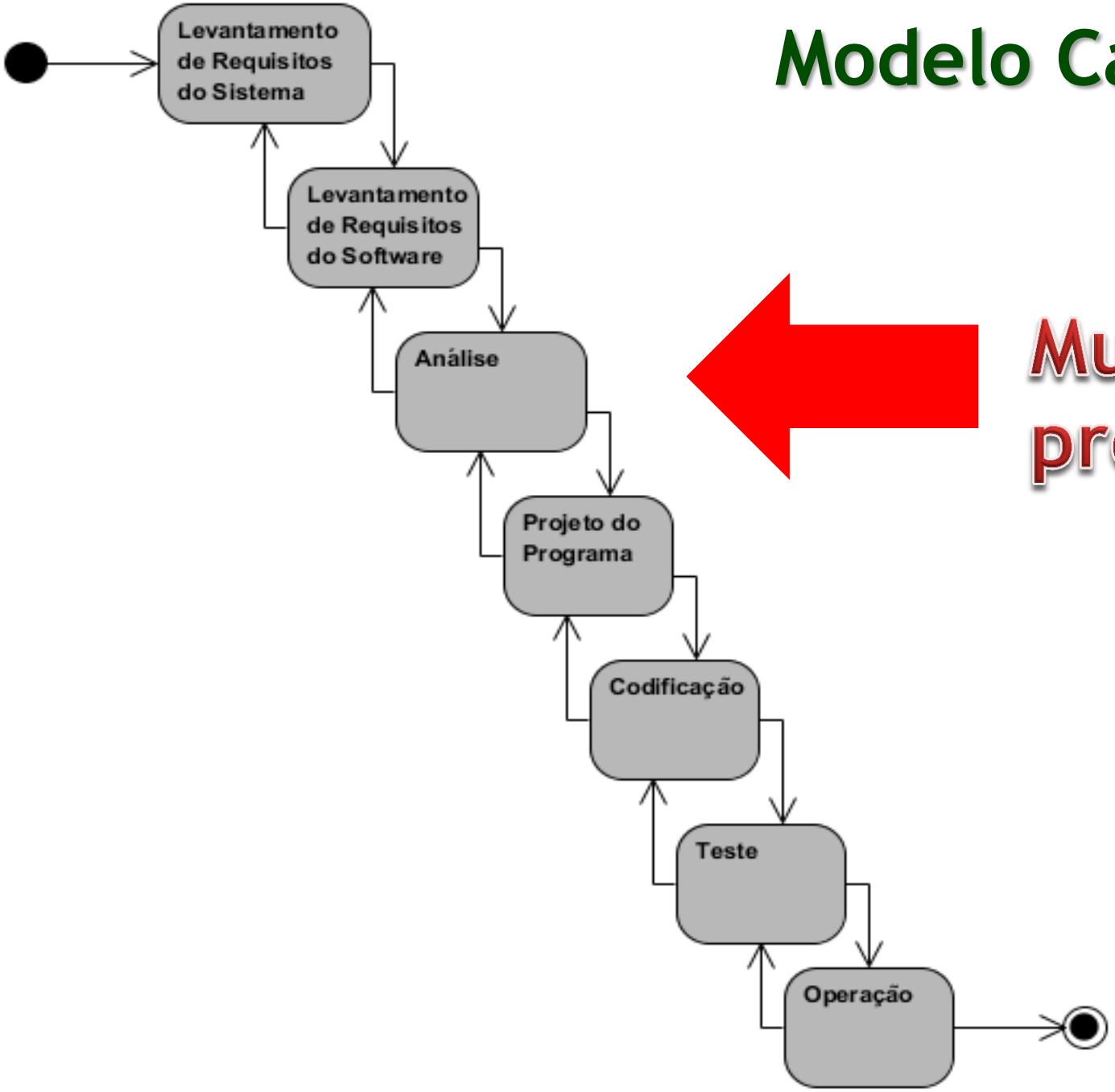
Métodos tradicionais X Métodos ágeis

Engenharia de Software I

Leitura

- **Livro**: Engenharia de Software: uma abordagem profissional - 7ª edição (2011) Autor: Pressman
- Cap 3. Desenvolvimento ágil

Modelo Cascata



Muitos problemas

Problemas do cascata

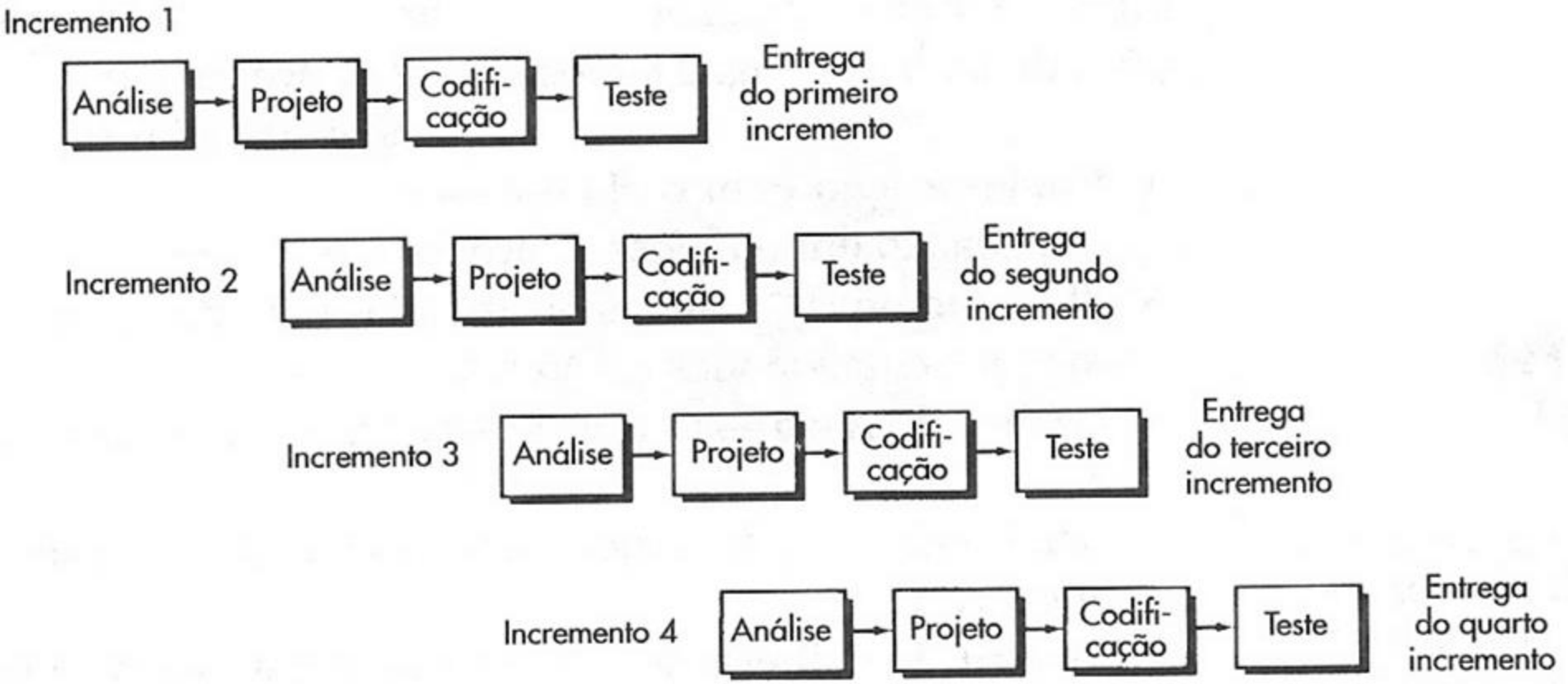
- Dificuldade de realizar um planejamento minucioso dos projetos
- Escopo mal definido: falta de visão clara do produto
- Mudança constante de requisitos: requisitos não são estáveis
- Falta de participação do cliente: o cliente normalmente assume a postura de ser o comprador do produto de software
- Comunicação falha: muitas fases, muitos documentos e muitos envolvidos

Introdução

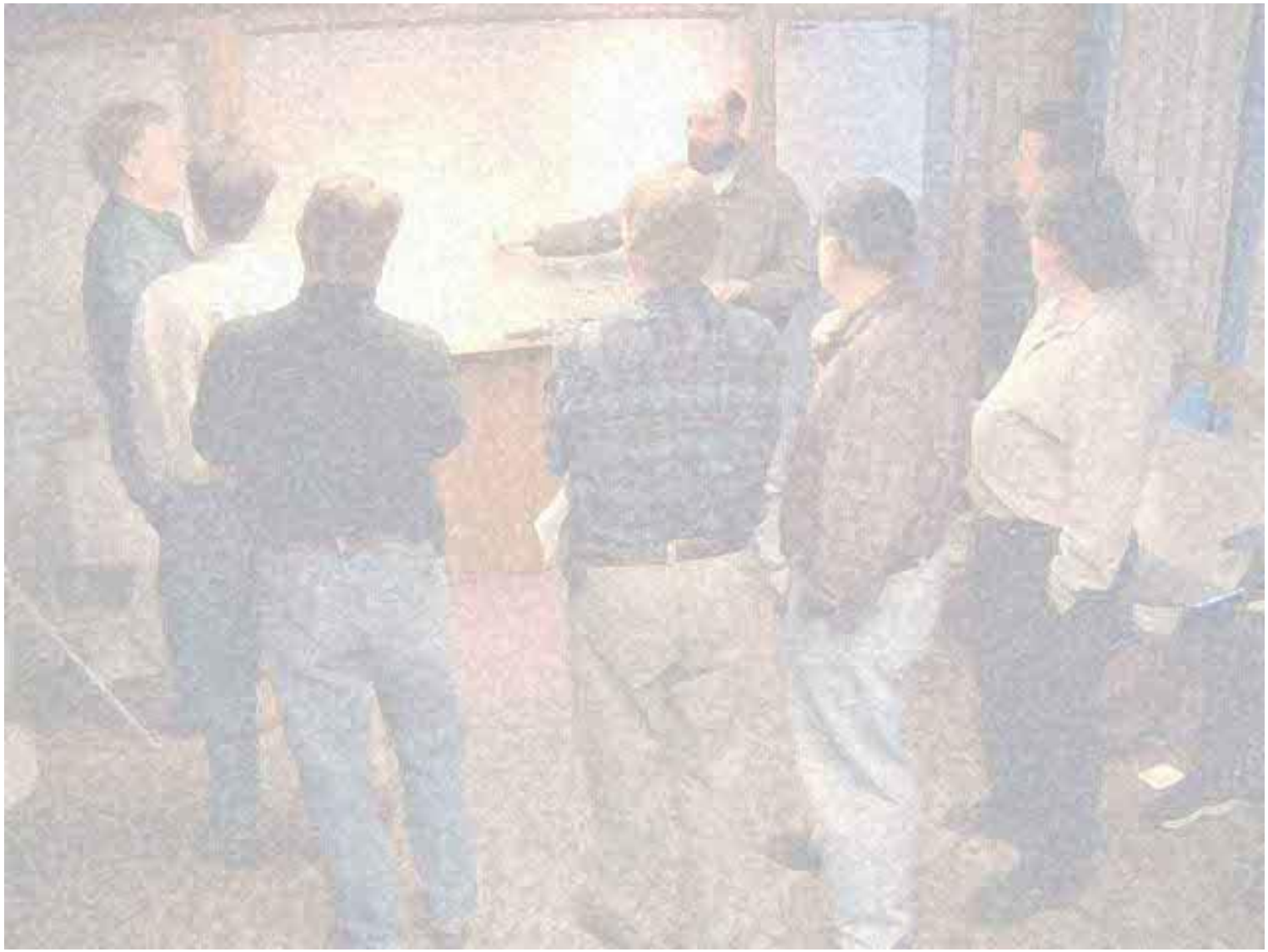
Situação atual das empresas de software:

- Estão buscando mais rapidez e produtividade
- Os projetos estão inseridos em ambientes de negócio bastante dinâmicos, sujeitos a mudanças constantes

Modelo incremental e iterativo

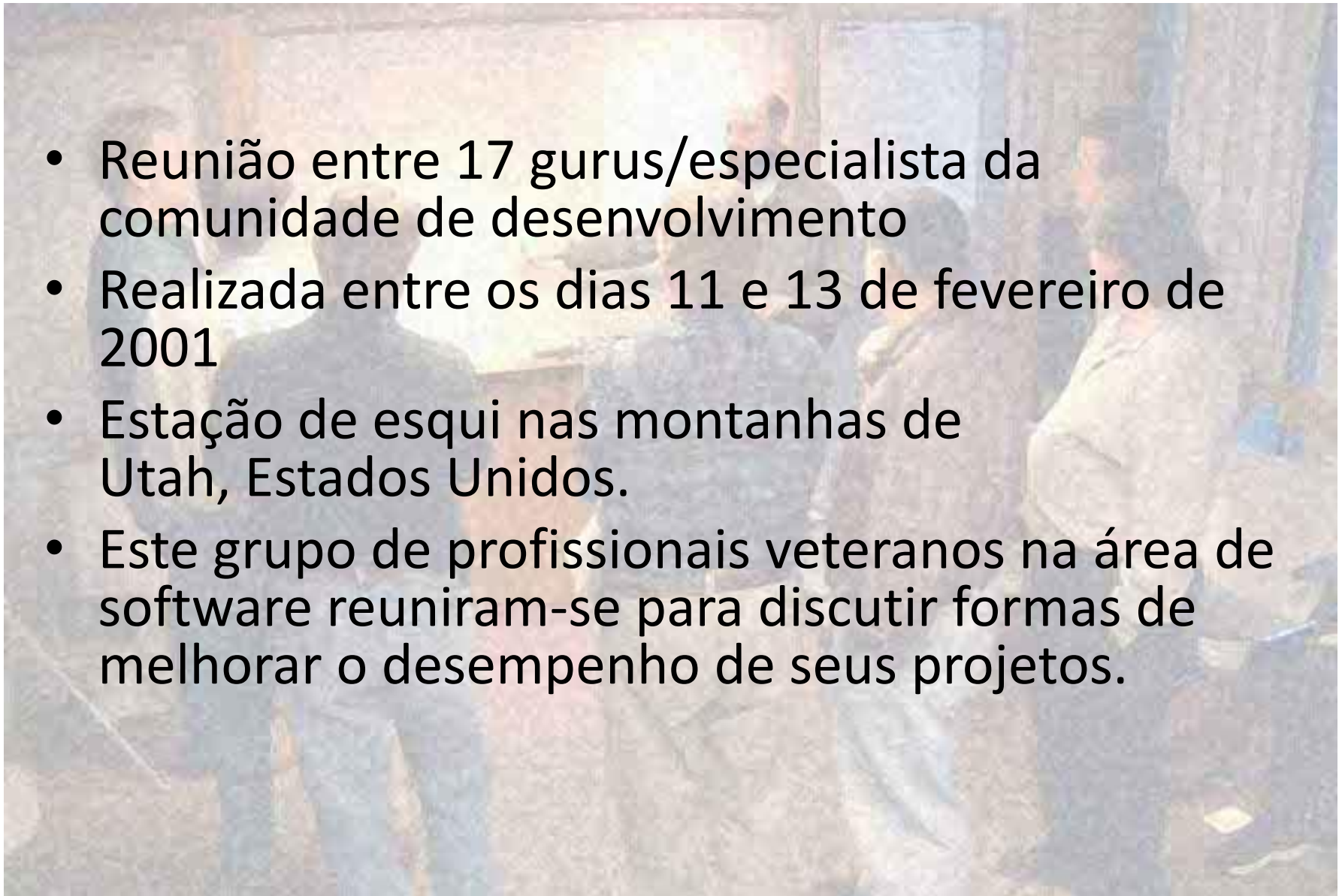


Manifesto ágil



Fonte da imagem: http://manifestoagil.com.br

Manifesto para o desenvolvimento Ágil de Software



- Reunião entre 17 gurus/especialista da comunidade de desenvolvimento
- Realizada entre os dias 11 e 13 de fevereiro de 2001
- Estação de esqui nas montanhas de Utah, Estados Unidos.
- Este grupo de profissionais veteranos na área de software reuniram-se para discutir formas de melhorar o desempenho de seus projetos.

Valores do Manifesto Ágil

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 funcionando** ➔ mais que documentações abrangente
- **Colaboração com o cliente** ➔ mais que negociação de contratos
- **Adaptação a mudanças** ➔ mais importante que seguir o plano inicial

Mesmo havendo valor nos itens à direita, valorizar mais os itens à esquerda.

Princípios do manifesto ágil

1) Satisfazer o cliente através da entrega contínua de software com valor agregado.

2) Mudanças nos requisitos são bem-vindas

3) Entregar frequentemente software funcionando

Princípios do manifesto ágil

4) Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.

5) Construa projetos em torno de indivíduos motivados.

6) Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho (conversa face a face).

Dyba e Dingsoyr (2009) afirmam:

"os membros da equipe tradicionais são menos substituíveis do que em equipes ágeis".

Dyba, T. Dingsoyr, T. What Do We Know about Agile Software Development?, **IEEE Software**, vol.26, n.5, p.6-9, Sept.-Oct. 2009.

Princípios do manifesto ágil

7) Software funcionando é a medida primária de progresso.

8) Os processos ágeis promovem desenvolvimento sustentável.
Todos devem manter ritmo constante indefinidamente.

9) Contínua atenção à excelência técnica e bom design aumenta a agilidade.

Princípios do manifesto ágil

10) Simplicidade é essencial → a arte de maximizar a quantidade de trabalho não realizado.

11) As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.

12) Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Métodos ágeis

Principais Métodos Ágeis:

- Scrum
- eXtreme Programming – XP
- Kanban
- Scrum + Kanban = Scrumban
- Lean Development Software

SCRUM

SCRUM

- No **jogo de rugby**, “**scrum**” é uma **estratégia** onde **todos os jogadores** do time **unem-se para atingir um objetivo**
- Devem ser retirados os obstáculos da frente dos jogador e correr com a bola, para que possa avançar em campo

Scrum

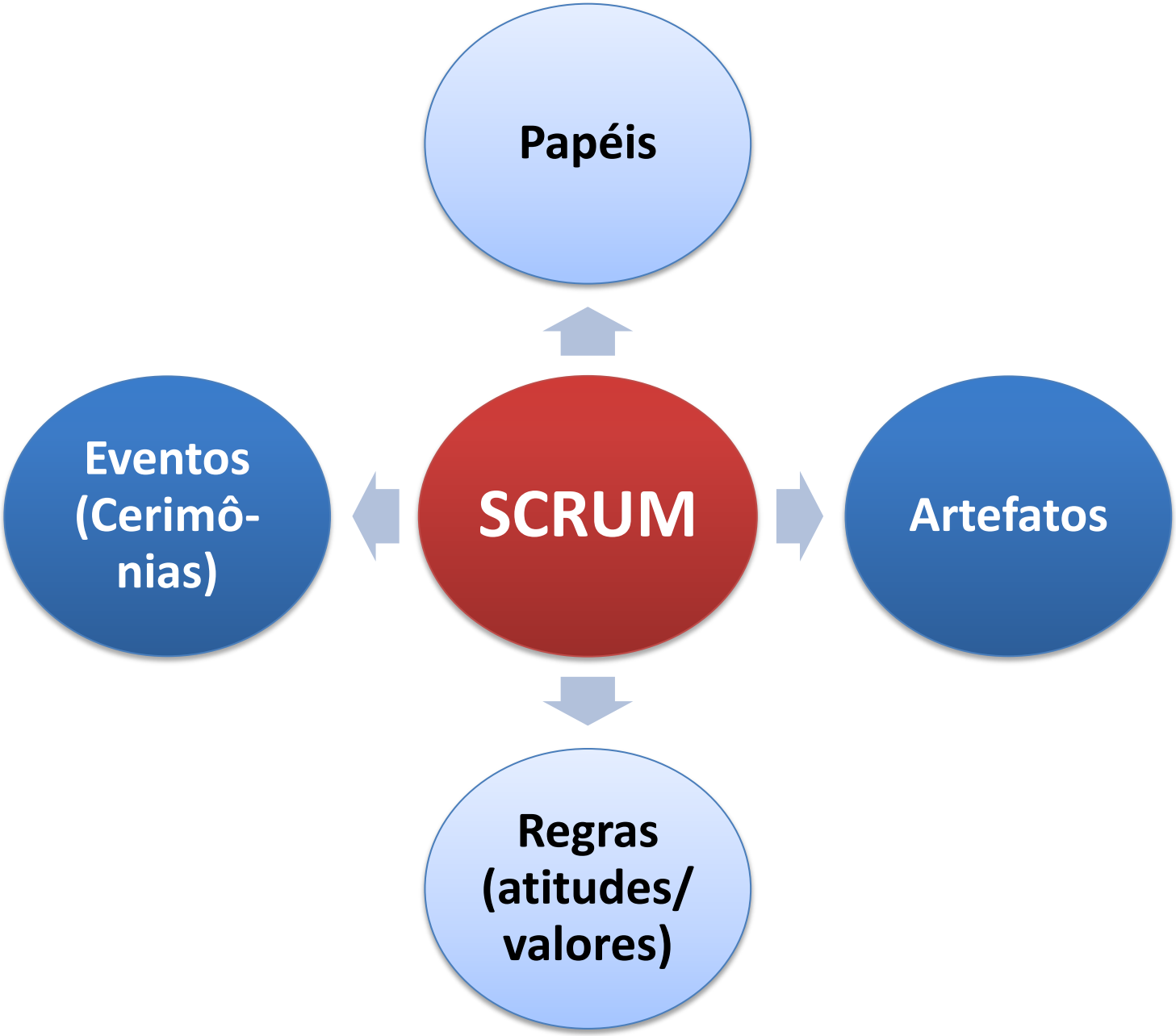
- É um método ágil de gerenciar projetos
- É um processo iterativo e incremental para desenvolvimento de produtos
- Scrum é mais atitude que processo
- Scrum **não** é: “Bala de prata”

Empresas que usam Scrum

- Google
- Globo
- HP
- BBC
- Amazon.com
- Nasa
- Instituto Nokia
- Salesforce.com
- Microsoft
- Datasul
- RBS
- CI&T
- Dígitro
- H2J
- Stefanini
- Vivo
- Petrobras
- Locaweb
- Gol
- Sul América Seguros
- SERPRO
- ..... e muitas outras

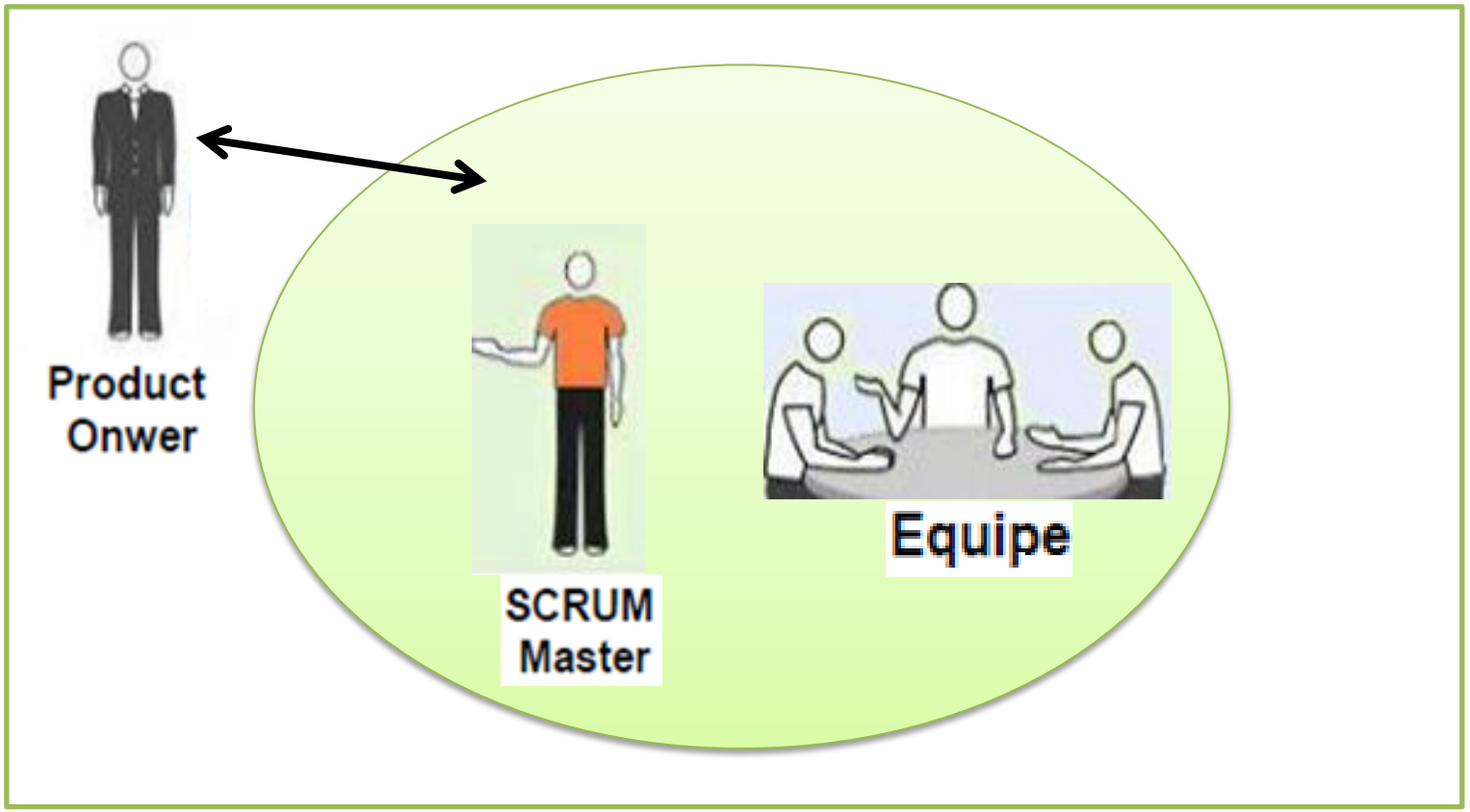
SCRUM

- **Scrum não é um método apenas para desenvolvimento** de software, **é um método de gestão de projetos** em geral.
- Scrum não é uma solução completa para os problemas de desenvolvimento de software
- **Segundo os autores**, **projetos com equipes pequenas** e **multidisciplinares** produzem **melhores resultados**.


Papéis
Eventos (Cerimô-nias)
SCRUM
Artefatos
Regras (atitudes/ valores)

Papéis do SCRUM

Cada equipe Scrum possui 3 papéis:



Papéis do Scrum
Product Owner (PO)

- ***Product Owner* (PO) → dono do produto ou representante do cliente**

Responsável por :

- Definir a visão do produto
- Elaborar , priorizar e manter o *Product Backlog* (objetivo do desenvolvimento, os requisitos a serem construídos e as prioridades)
- Aceitar ou rejeitar os entregáveis
- Assume a responsabilidade do projeto.

Product Onwer

Papéis do SCRUM
Scrum Master

- ***O ScrumMaster*** → **responsável por garantir que o processo seja compreendido e seguido**

Responsável por:

- Remover impedimentos
- Proteger a equipe
- Ajudar o PO (quando necessário)
- Ser o facilitador da equipe
- Responsável pelo sucesso do Scrum: garantir que todos sigam as regras do scrum

Papéis do SCRUM
Equipe / Time

- **A *equipe (ou time)* → responsável pelo desenvolvimento do produto**


Equipe

- A Equipe ainda é responsável por:
  - Fazer estimativa;
  - Definir as tarefas;
  - Garantir a qualidade do produto;
  - Apresentar o produto ao cliente.
- Auto-organizado e auto-gerenciado
- O tamanho ótimo para uma equipe é de 7 pessoas, mais ou menos duas pessoas, não sendo o PO e o *Scrum Master*.
- A equipe deve ser formada por pessoas comprometidas e ter habilidades necessários para atingir as metas das *Sprints*

Papéis do Scrum e Equipe

Para trabalhar como o SCRUM é preciso **trabalhar em equipe** a qual deverá ter as seguintes características (5):

1) ter **Auto gestão**;

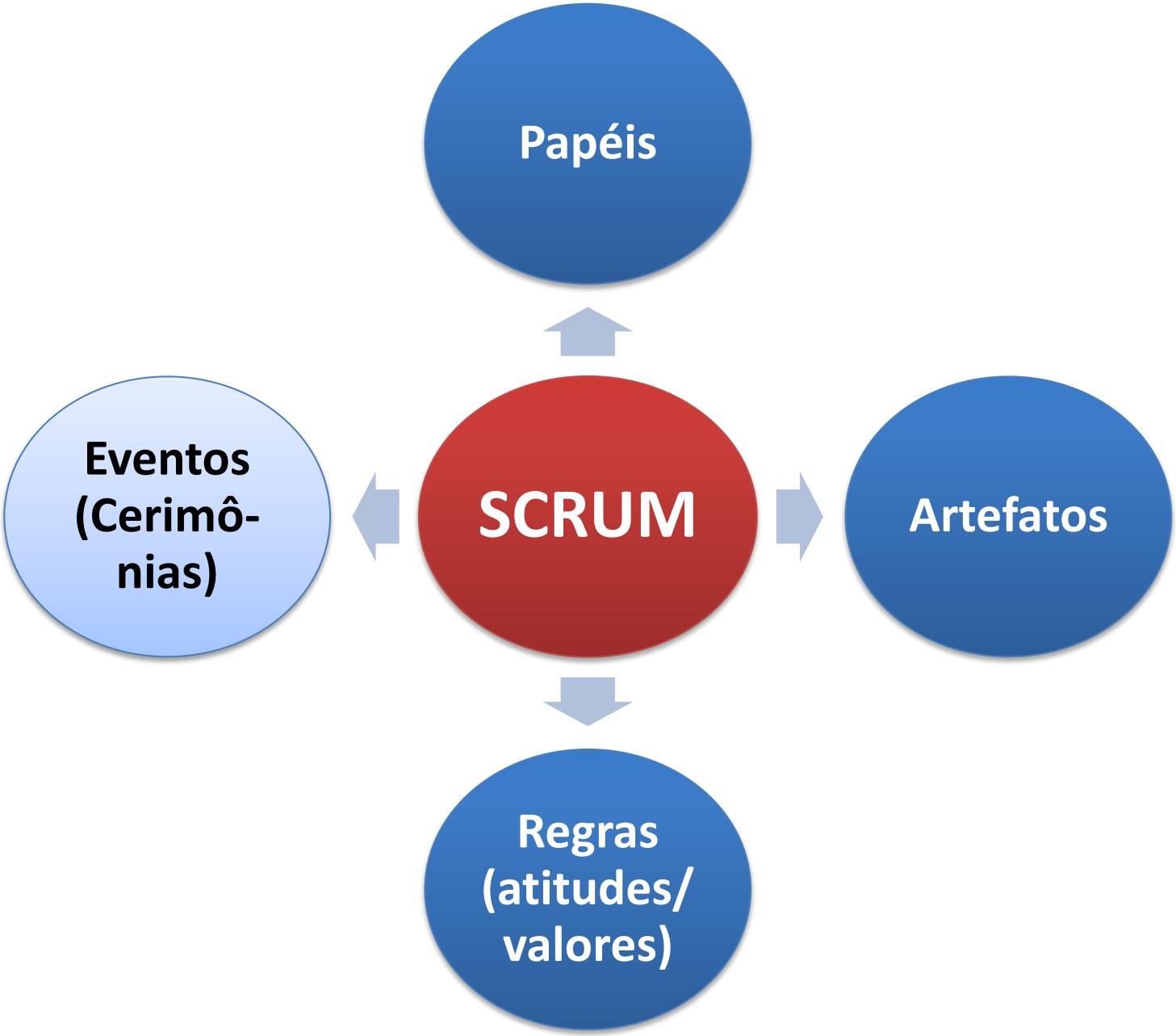
   A auto gestão: Requer **alto comprometimento** da equipe, que é a chave para a produtividade. Equipe motivada produz mais e melhor.

2) ser **Auto organizada**;

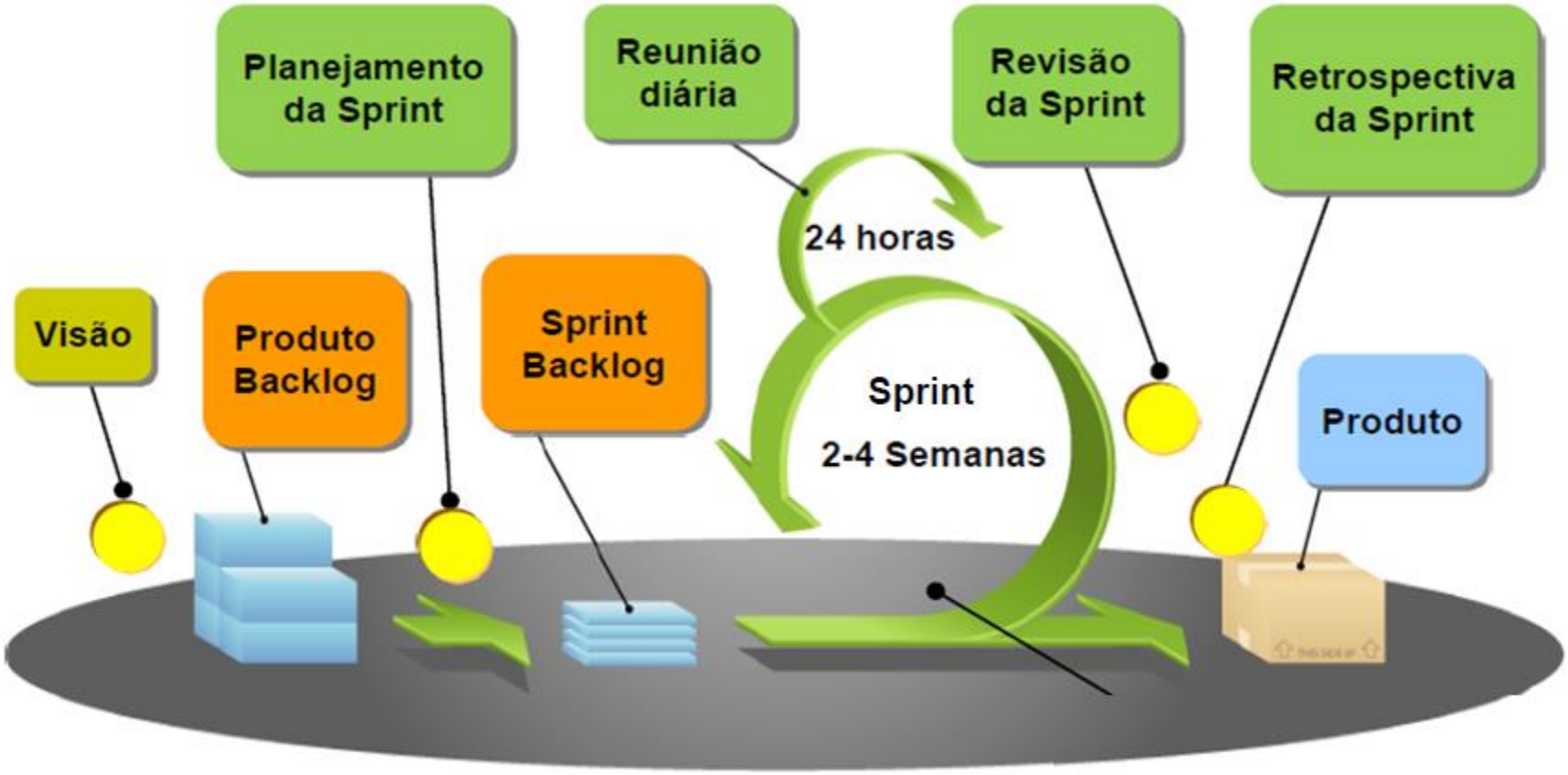
3) ser **Interdisciplinar**: os membros da equipe devem ter todo o conhecimento e habilidades necessárias para entregar a meta da Sprint.

4) **não ter Hierarquia formal**,

5) ter **Responsabilidade**.


Papéis
Eventos (Cerimô-nias)
SCRUM
Artefatos
Regras (atitudes/ valores)

Fluxo do SCRUM



Legenda:

Reuniões

Artefatos

Eventos (Reuniões)

- Planejamento da Sprint
- Diária
- Revisão da Sprint
- Retrospectiva da Sprint

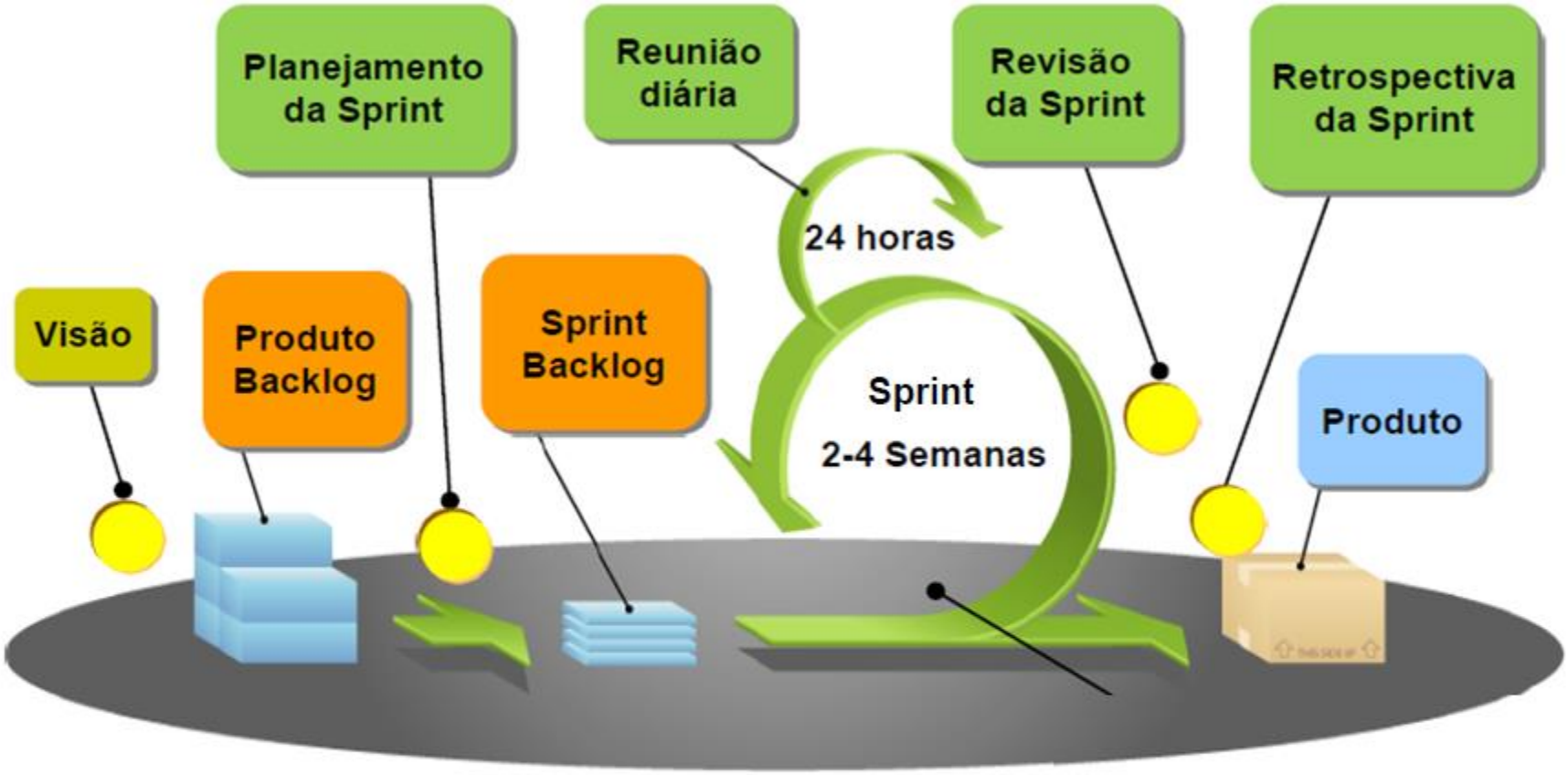
Artefatos

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

Sprint

- **A Sprint é a parte central do Scrum**
- É uma iteração que deve ser realizada de 2 a 4 semanas. Estão reduzindo para 1 semana.
- Neste tempo a equipe deve produzir software para ser entregue ao cliente
  - O time recebe uma parte do *product backlog* para desenvolvimento
  - O *product* backlog não sofrerá modificações durante o Sprint
- Cada Sprint gera um novo incremento para o produto final

Fluxo do SCRUM



Legenda:

Reuniões

Artefatos

Eventos (Reuniões)

- Planejamento da Sprint
- Diária
- Revisão da Sprint
- Retrospectiva da Sprint

Artefatos

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

Reunião diária

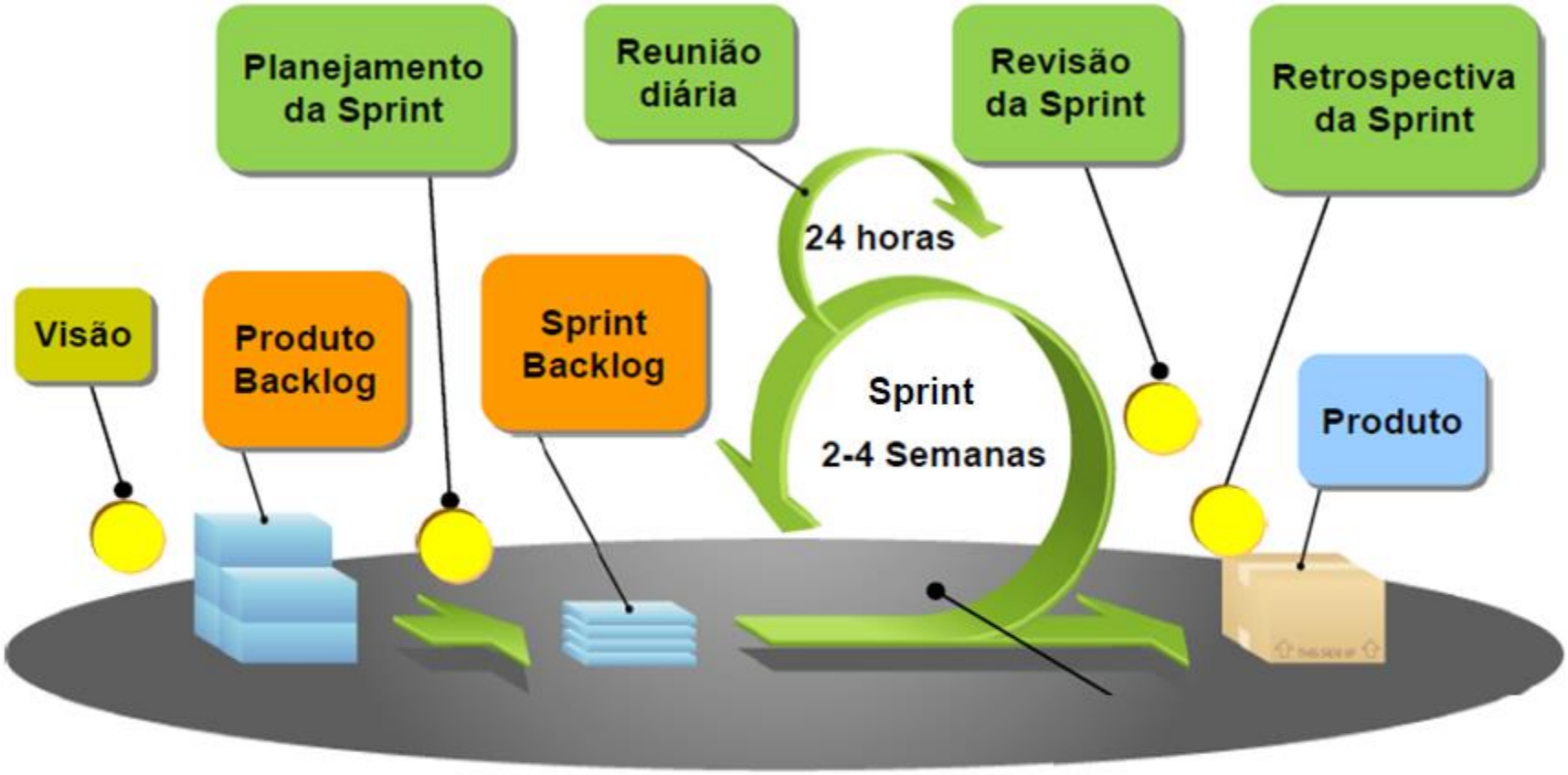
- Reunião de 15 minutos
- **Todos devem responder as perguntas:**
  - **O que você realizou desde a última reunião?**
  - **Quais problemas/impedimentos você enfrentou?**
  - **Em que você irá trabalhar até a próxima reunião?**
- Benefícios:
  - Maior integração entre os membros da equipe
  - Rápida solução de problemas
  - Progresso medido continuamente
  - Promove o compartilhamento do conhecimento

Reunião diária

Importante:

- Todos devem participar
- Não é pausa para o lanche
- Não é brincadeira
- Deve ser mais objetiva possível, ou seja, deve se ater as 3 perguntas e nada mais

Fluxo do SCRUM



Legenda:

Reuniões

Artefatos

Eventos (Reuniões)

- Planejamento da Sprint
- Diária
- Revisão da Sprint
- Retrospectiva da Sprint

Artefatos

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

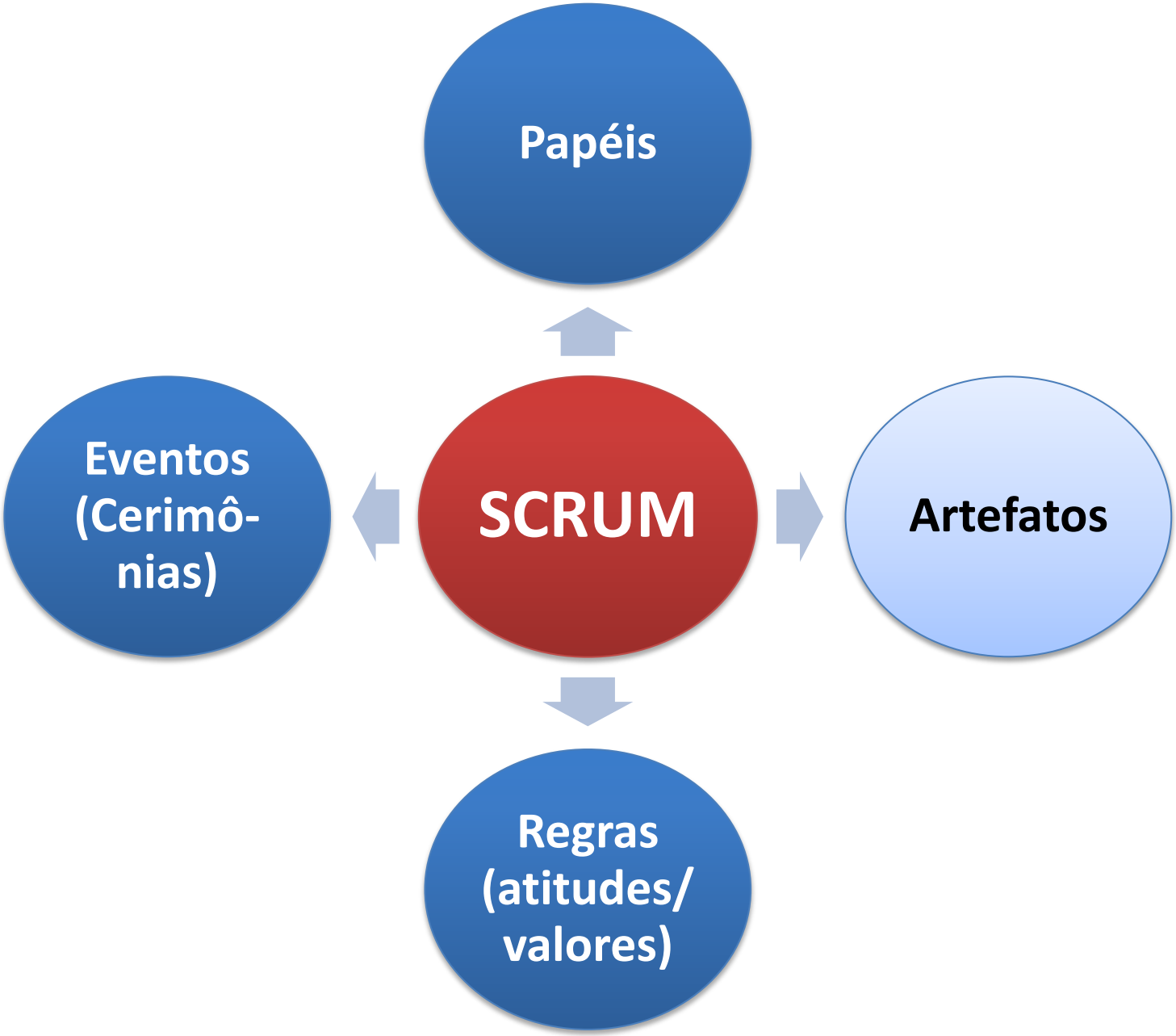
Review (revisão) da Sprint

- Reunião onde o time irá apresentar o resultado da Sprint para o PO e seus convidados
- Deve obedecer à data de entrega
- Deverão ser apresentados apenas os itens que estiverem 100% prontos
- Deve haver uma demonstração funcional do que foi produzido na Sprint
- Benefícios:
  - Apresentar resultados concretos ao cliente
  - Integrar e testar uma boa parte do software
  - Motivação da equipe

Reunião de retrospectiva da Sprint

- Objetivo identificar o que teve de bom e o que deve ser melhorado em uma Sprint, ou seja, garantir o processo de melhoria continua
- Os membros do time devem responder a duas questões:
  - **O que aconteceu de bom durante o último *Sprint*?**
  - **O que pode ser melhorado para o próximo *Sprint*?**

- *ScrumMaster* escreve as respostas e prioriza na ordem que deseja discutir as potenciais melhorias


Papéis
Eventos (Cerimô-nias)
SCRUM
Artefatos
Regras (atitudes/ valores)

Product Backlog

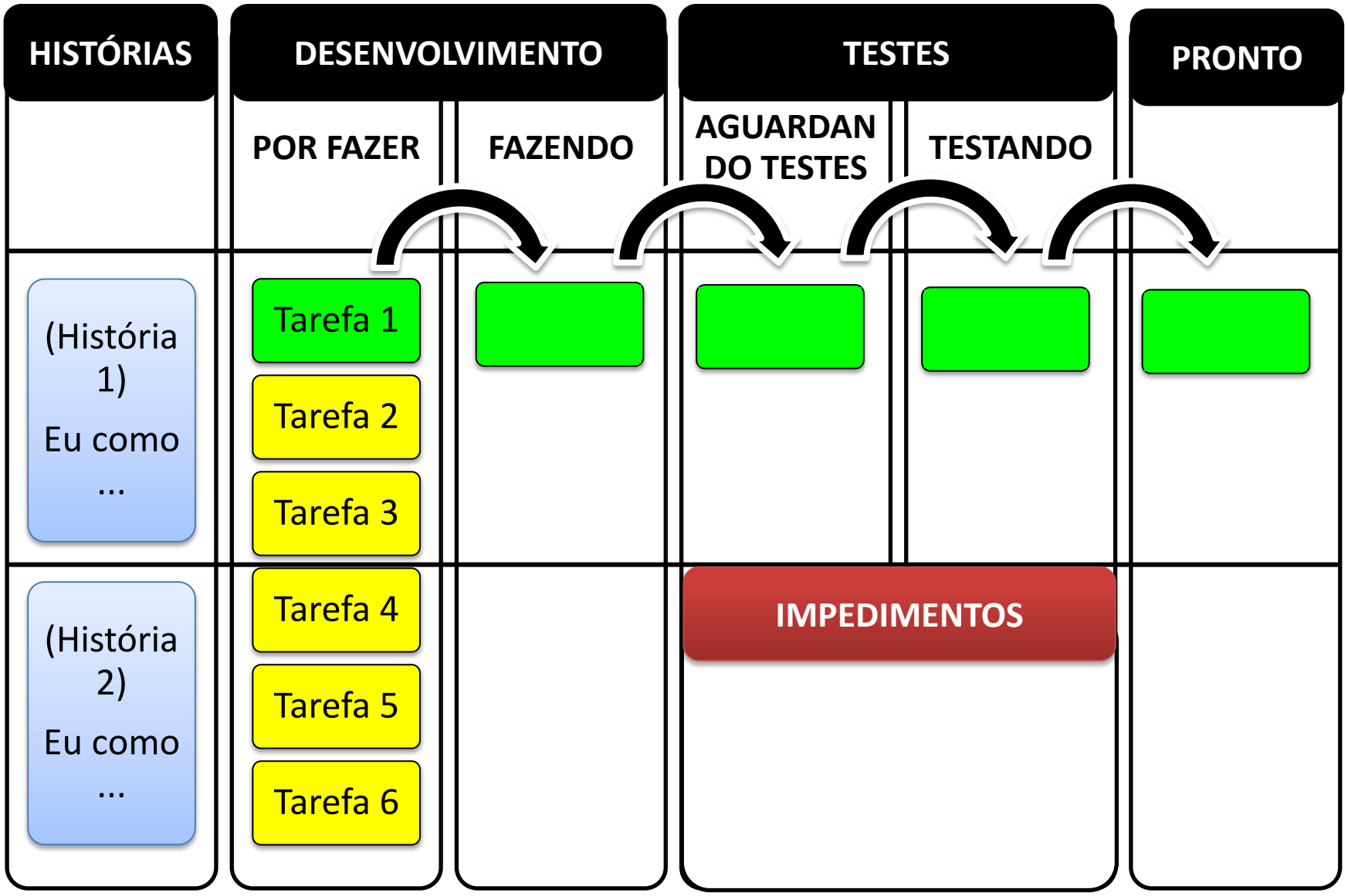
- Documento que representa a visão do produto de forma modular
- Deve estar contida nele todos os itens que deverão ser desenvolvidos durante o projeto
- Deve ser descrito de forma clara e simples, de forma que seja de fácil entendimento tanto para o time de desenvolvimento quanto para o cliente
- Escrita normalmente em forma de histórias de usuários

Quadro Kanban

Itens	Tarefas	Em Andamento	Pronto
Item 1	Tar 1, Tar 2, Tar 3, Tar 4		
Item 2	Tar 1, Tar 2, Tar 3, Tar 4	Tar 5, Tar 6, Tar 9	Tar 8
Item 3		Tar 5, Tar 6, Tar 9	Tar 2, Tar 4, Tar 8, Tar 1, Tar 3

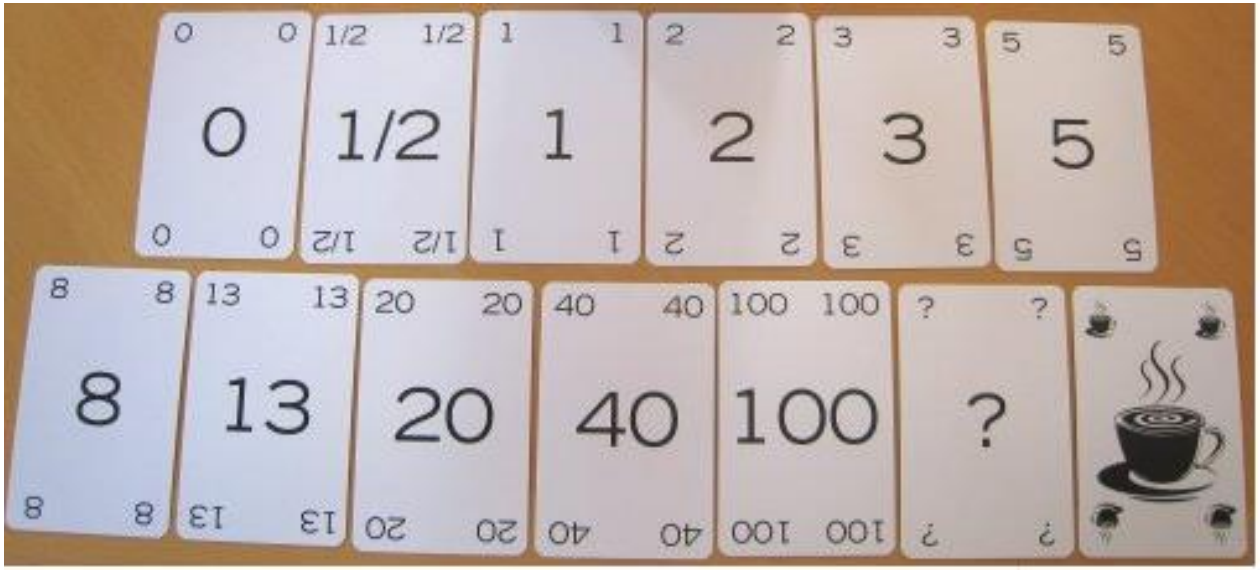
Quadro Kanban

- Para maior visualização do time as tarefas a serem realizadas na Sprint podem ser apresentadas em forma de quadro Kanban
- Kanban é uma ferramenta visual que serve para sinalizar o estado atual de projetos Scrum
- Tem como principal objetivo dar maior visibilidade ao andamento do desenvolvimento do projeto
- Deve estar visível e de fácil acesso a todos os membros do time

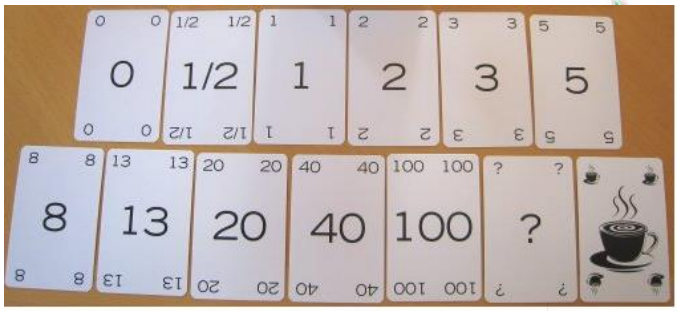

HISTÓRIAS
DESENVOLVIMENTO
TESTES
PRONTO
POR FAZER
FAZENDO
AGUARDANDO TESTES
TESTANDO
(História 1)
Eu como ...
Tarefa 1
Tarefa 2
Tarefa 3
(História 2)
Eu como ...
Tarefa 4
Tarefa 5
Tarefa 6
IMPEDIMENTOS

Estimativas com Planning Poker

- As cartas:
  - Sequencia de Fibonacci: 1,2,3,5,8,13,.....
  - Carta 0: isso já está pronto
  - Carta ?: dúvida sobre o que fazer
  - Carta café: vamos dar uma pausa



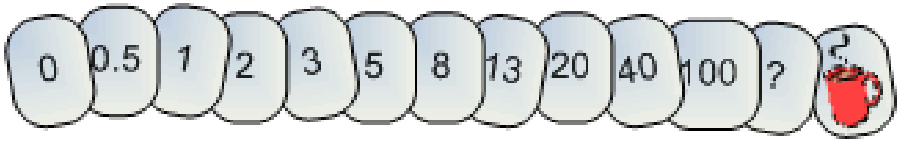
Estimativas com Planning Poker



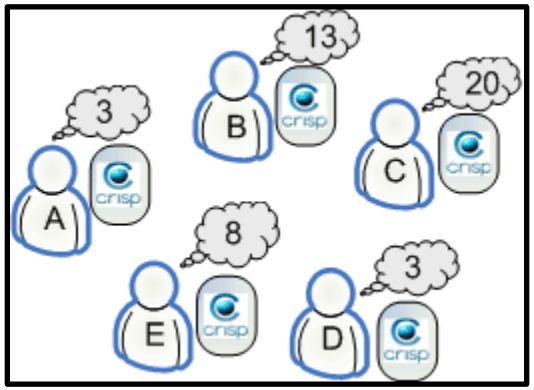
- Técnica de estimativa muito utilizada nos métodos ágeis
- Todas pessoas do time de desenvolvimento participam da estimativa
  - visa o comprometimento dos membros da time
  - Todos são responsáveis pela sua concretização da execução conforme a estimativa realizada
- Deve haver um consenso
- Evita/minimiza a influência de alguns membros do time

Planning Poker

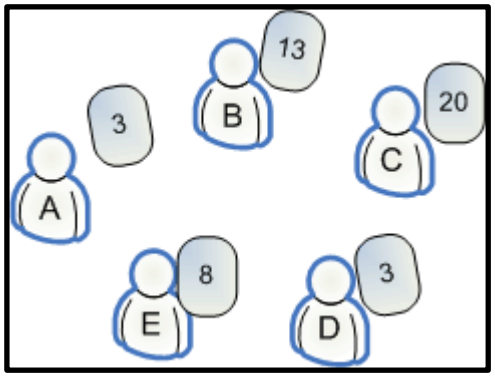
- Exemplo



1



2



3

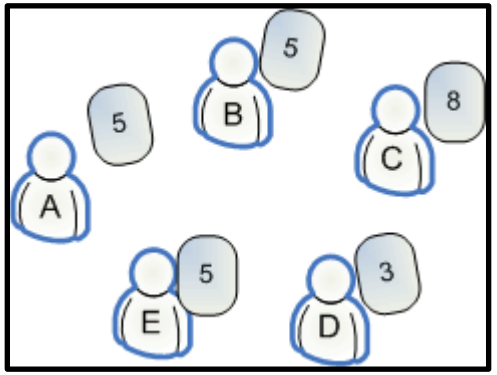
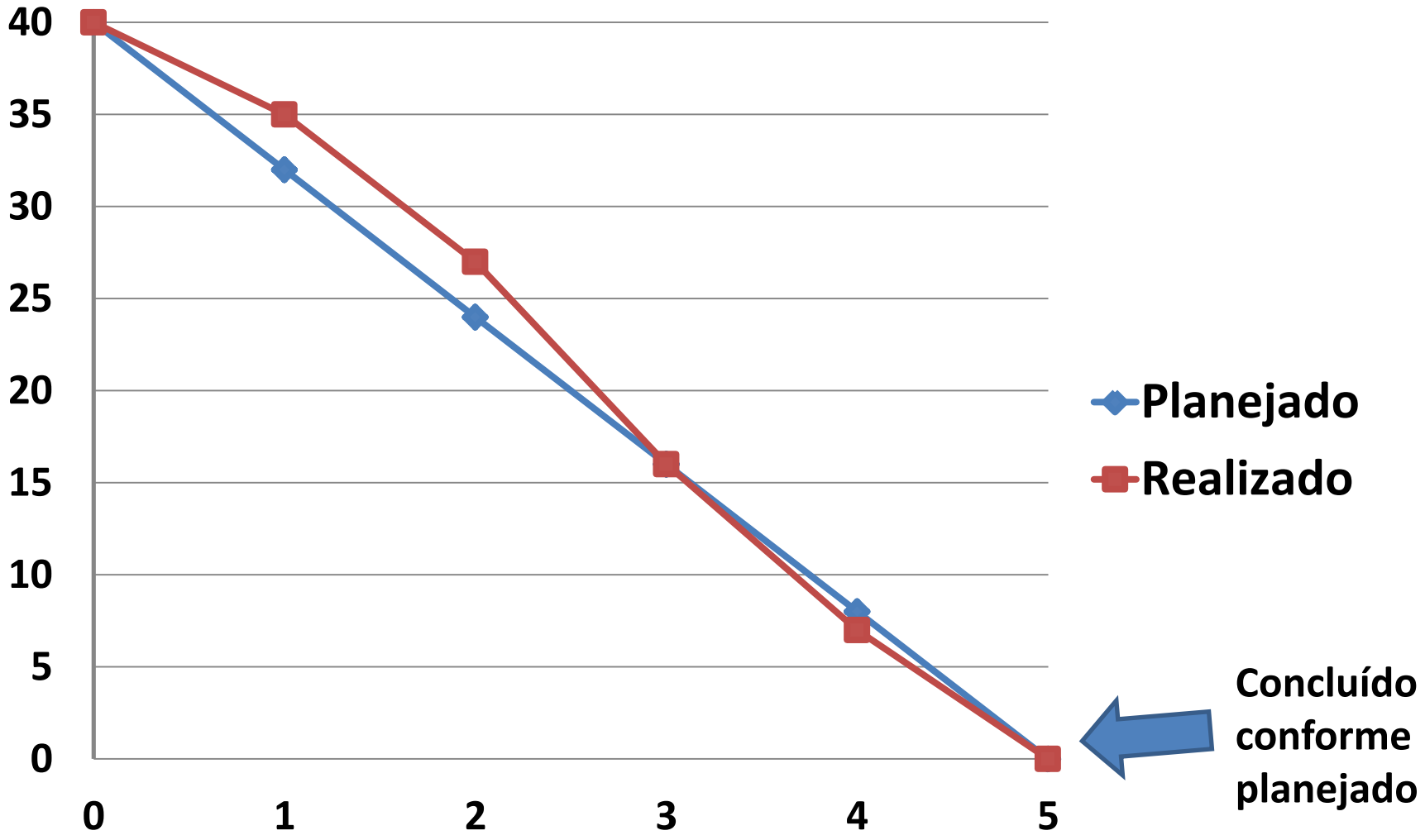


Gráfico Burndown - Dia 5



Total de horas da sprint: 40

Pontos previstos no 5º dia: 8 → acumulado previsto: 8-8=0

Soma do trabalho realizado no 5º dia: 7

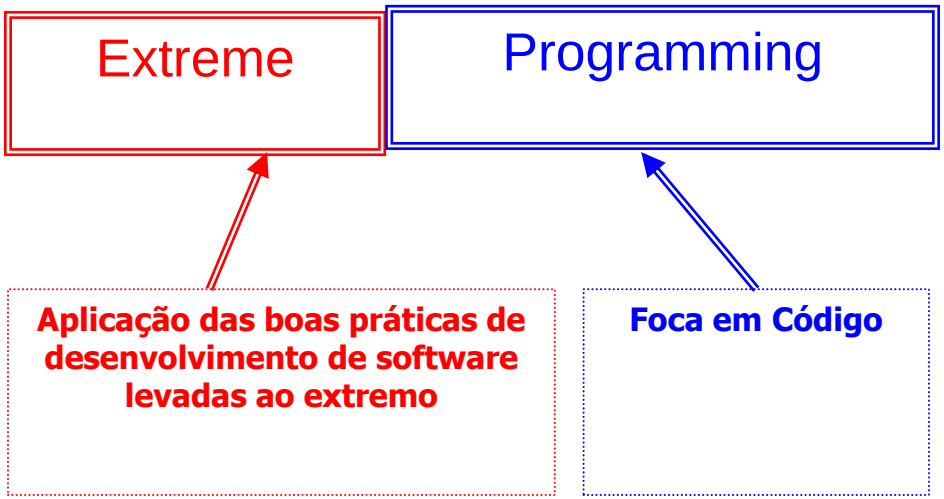
Ponto do gráfico burndown no 5º dia: 7-7=0

Burndown

eXtreme Programming
XP

eXtreme Programming - XP

- Desenvolvido por kent Beck
- Desenvolvida para:
  - Equipes médias e pequenas (2 a 12 pessoas)
  - Requisitos vagos e em constante evolução
- Possui um conjunto de valores e práticas para nortear o desenvolvimento de software

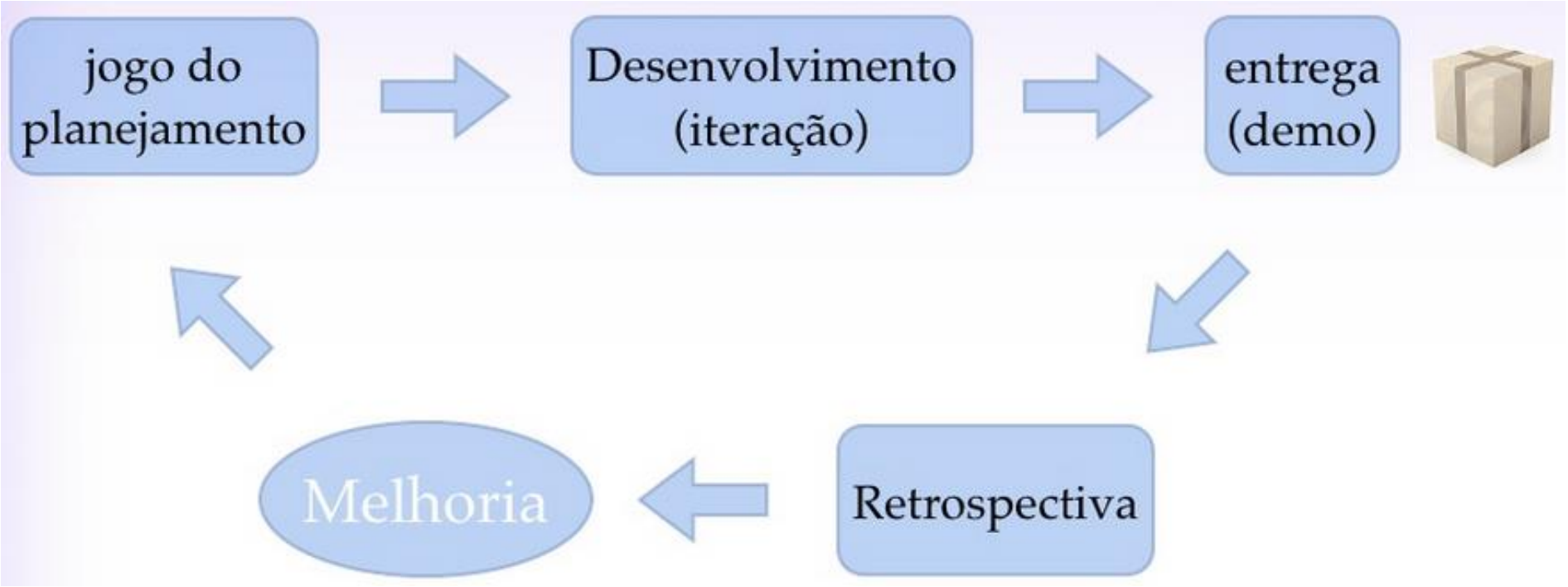


Levar as boas práticas ao extremo:

- Se testar é bom, vamos testar toda hora
- Se projetar é bom, vamos fazer disso parte do trabalho diário
- Se integrar é bom, vamos integrar o máximo possível
- Se iterações curtas é bom, vamos deixar as iterações realmente curtas

➔ **“Xp é mudanças social” (Kent Beck)**

Ciclo de vida do XP



XP - Princípios básicos

- **Comunicação**
  - Os membros da equipe (clientes, gerentes, programadores) devem interagir ao máximo pessoalmente.
  - Devem trabalhar na mesma sala, conversar pessoalmente ou através de *chats*, etc.
- **Simplicidade**
  - Projeto do SW é simplificado continuamente
  - Processo adaptado, caso algo não esteja funcionando
- **Feedback**
  - Cliente está sempre participando do desenvolvimento do sistema
  - Testes de unidade e de aceitação fornecem feedback sobre o sistema
  - Oportunidades e problemas são identificados o mais rápido possível
- **Coragem**
  - Indicar problemas no projeto, parar quando está cansado, pedir ajuda quando necessário
  - Simplificar código que está funcionado, dizer ao cliente que não será possível implementar um requisito no prazo estimado
  - Seguir o XP como deve ser

XP - Práticas

- Jogo do Planejamento
- Versões Pequenas
- Metáfora
- Projeto Simples
- Desenvolvimento Orientado por Testes (TDD)
- Testes dos Clientes
- Refatoração
- Programação Pareada
- Propriedade Coletiva do Código
- Integração Contínua e Frequente
- Ritmo Sustentável
- Cliente com os desenvolvedores
- Padrão de Código

XP

- **Jogo de Planejamento (Planning Game):**
  - O desenvolvimento é feito em iterações semanais.
  - No início da semana, desenvolvedores e cliente reúnem-se para priorizar as funcionalidades.
  - Essa reunião recebe o nome de Jogo do Planejamento.
  - Nela, o cliente identifica prioridades e os desenvolvedores as estimam.
  - O cliente é essencial neste processo e assim ele fica sabendo o que está acontecendo e o que vai acontecer no projeto.
  - Como o escopo é reavaliado semanalmente, o projeto é regido por um contrato de escopo negociável, que difere significativamente das formas tradicionais de contratação de projetos de software.
  - Ao final de cada semana, o cliente recebe novas funcionalidades, completamente testadas e prontas para serem postas em produção.
- **Pequenas Versões (Small Releases):**
  - A liberação de pequenas versões funcionais do projeto auxilia muito no processo de aceitação por parte do cliente, que já pode testar uma parte do sistema que está comprando.

XP

- **Metáfora (Metaphor):**
  - Procura facilitar a comunicação com o cliente, entendendo a realidade dele.
  - O conceito de rápido ou processo para um cliente de um sistema jurídico é diferente para um programador experiente em controlar comunicação em sistemas em tempo real, como controle de tráfego aéreo.
  - É preciso traduzir as palavras do cliente para o significado que ele espera dentro do projeto.
- **Projeto Simples (Simple Design):**
  - Simplicidade é um princípio da XP.
  - Um erro comum ao adotar essa prática é a confusão por parte dos programadores de código simples e código fácil.
  - Nem sempre o código mais fácil de ser desenvolvido levará a solução mais simples por parte de projeto.
  - Esse entendimento é fundamental para o bom andamento do XP.
  - Código fácil deve ser identificado e substituído por código simples.

XP

- **Testes de Aceitação (Customer Tests):**
  - São testes construídos pelo cliente e conjunto de analistas e testadores, para aceitar um determinado requisito do sistema.
- **Ritmo Sustentável (Sustainable Pace):**
  - Trabalhar com qualidade, buscando ter ritmo de trabalho saudável (40 horas/semana, 8 horas/dia), sem horas extras.
  - Horas extras são permitidas quando trouxerem produtividade para a execução do projeto.
  - Outra prática que se verifica neste processo é a prática de trabalho energizado, onde se busca trabalho motivado sempre.
  - Para isto o ambiente de trabalho e a motivação da equipe devem estar sempre em harmonia.
- **Reuniões em pé (Stand-up Meeting):**
  - Reuniões em pé para não se perder o foco nos assuntos, produzindo reuniões rápidas, apenas abordando tarefas realizadas e tarefas a realizar pela equipe.

XP

- **Programação em Pares (Pair Programming):**
  - É a programação em par/dupla num único computador.
  - Geralmente a dupla é formada por um iniciante na linguagem e outra pessoa funcionando como um instrutor.
  - Como é apenas um computador, o novato é que fica à frente fazendo a codificação, e o instrutor acompanha ajudando a desenvolver suas habilidades.
  - Desta forma o programa sempre é revisto por duas pessoas, evitando e diminuindo assim a possibilidade de defeitos.
  - Com isto busca-se sempre a evolução da equipe, melhorando a qualidade do código fonte gerado.

XP

- **Padrões de Codificação (Coding Standards):**
  - A equipe de desenvolvimento precisa estabelecer regras para programar e todos devem seguir estas regras.
  - Desta forma parecerá que todo o código fonte foi editado pela mesma pessoa, mesmo quando a equipe possui 10 ou mais membros.
- **Refatoração (Refactoring):**
  - É um processo que permite a melhoria continua da programação, com o mínimo de introdução de erros e mantendo a compatibilidade com o código já existente.
  - Refatoração melhora a clareza (leitura) do código, divide-o em módulos mais coesos e de maior reaproveitamento, evitando a duplicação de código-fonte;
- **Integração Contínua (Continuous Integration):**
  - Sempre que produzir uma nova funcionalidade, nunca esperar uma semana para integrar à versão atual do sistema.
  - Isto só aumenta a possibilidade de conflitos e a possibilidade de erros no código fonte.
  - Integrar de forma contínua permite saber o status real da programação.