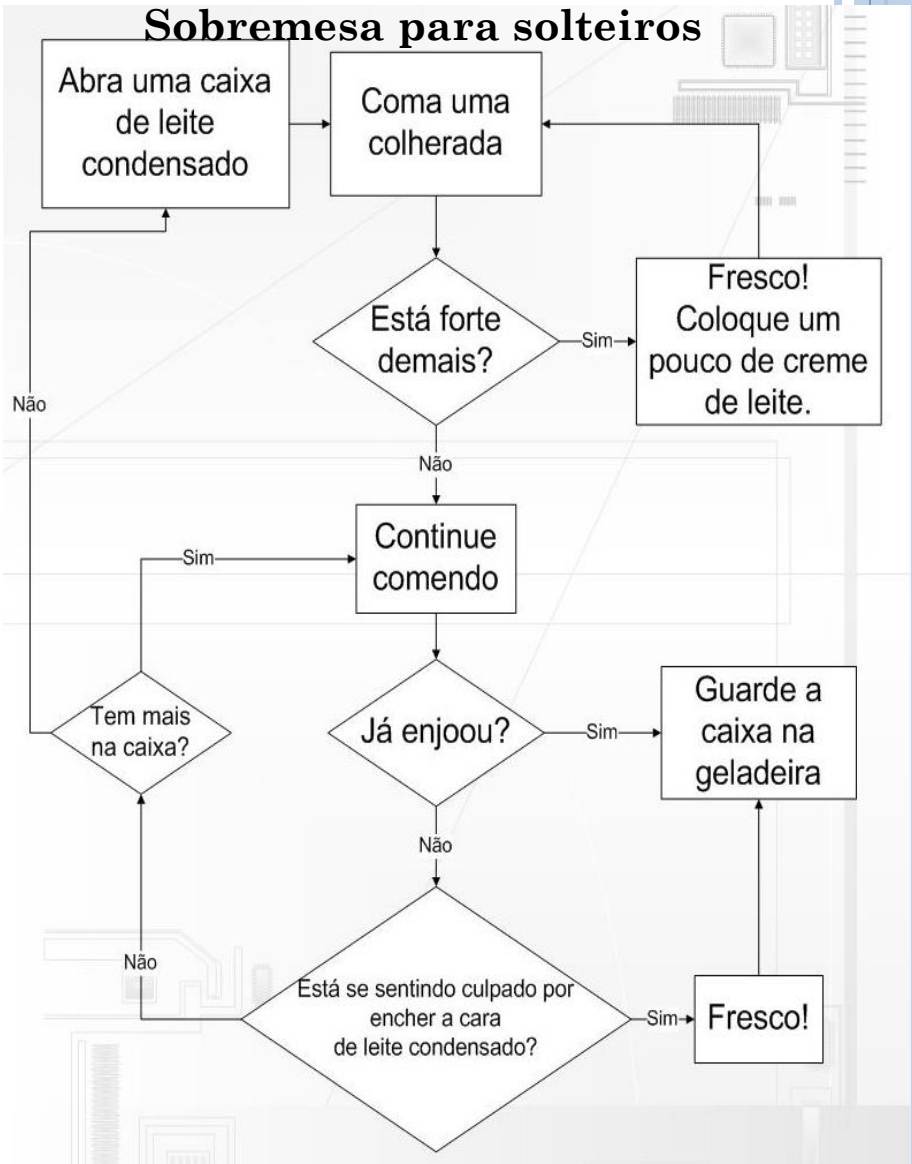
PROCESSO E CICLO DE VIDA DE SOFTWARE

Profa. Ana Flávia

PROCESSO?

- No latim *procedere* é verbo que **indica a ação de avançar, ir para frente** (pro+cedere)
- Conjunto de **manipulações para obter um resultado**.
- **Modo de fazer alguma coisa**.



SOFTWARE?

Software não é apenas o programa, mas também todos os dados de documentação e configuração associados, necessários para que o programa opere corretamente. *(Sommerville, 2008)*

REGRAS DO NEGÓCIO
MODELOS
PROJETOS
CÓDIGO FONTE
ARQUIVOS DE CONFIGURAÇÃO
AJUDA AO USUÁRIO
.....

+

```
53. // faz saque tendo como parâmetro o numero da conta
54. function saque($conta, $valor) {
55.
56.     // verifica se a conta existe
57.     if ($this -> conta == $conta) {
58.         // verifica se o saldo é suficiente
59.         if ($valor > $this -> valor) {
60.
61.             echo "Saldo Insuficiente.";
62.
63.         }
```

+

ENGENHARIA DE SOFTWARE

É uma disciplina de engenharia relacionada com todos os aspectos da produção de software. *(Sommerville, 2008)*

Como construir um software de qualidade.

ENGENHARIA DE SOFTWARE

- Segundo Pressman, a engenharia de software é uma **tecnologia em camadas** cuja pedra fundamental é o **foco na qualidade**.

- **Processo** é a liga que mantém as camadas de tecnologia **coesas**. Define uma metodologia e constitui a **base para o controle do gerenciamento** dos projetos de software

PROCESSO DE SOFTWARE?

"Combinação de atividades, ferramentas e procedimentos visando o desenvolvimento ou evolução de um software". *(Sommerville, 2011)*

"Roteiro, ou conjunto de passos, previsível que ajuda a criar a tempo um software de alta qualidade (*Pressman, 2011*)

PROCESSO DE SOFTWARE?

- Um processo de software
  - prescreve a ordem e a frequência de cada fase/atividade
  - especifica critérios para mudar de uma fase para outra
  - define o que tem que ser entregue ao final de cada fase

- Um processo de software NÃO significa
  - "sobrecarga", "papelada desnecessária", "perda da tempo"

- Um processo de software tem efeito positivo
  - para atender ao cronograma e obter software com mais qualidade e mais fácil de manter

PROCESSO DE SOFTWARE?

- Como escolher um processo
  - As **CARACTERÍSTICAS DA APLICAÇÃO** (domínio do problema, tamanho, complexidade etc);
  - A **TECNOLOGIA** a ser adotada na sua construção (paradigma de desenvolvimento, linguagem de programação, etc), a organização;
  - **ONDE** o produto será desenvolvido;
  - O **PERFIL DA EQUIPE** de desenvolvimento.

Quando se escolhe um processo define-se um MODELO DE PROCESSO (CICLO DE VIDA).

CICLO DE VIDA

"É uma representação abstrata de um processo de software. Cada modelo de processo representa um processo sob determinada perspectiva e, desta forma, fornece somente informações parciais sobre esse processo". *(Sommerville, 2008)*

CICLO DE VIDA

- Sugerem um roteiro de atividades, ações, tarefas, marcos e produtos de trabalho necessários para desenvolver um software com qualidade.

- Engenheiros de software e gerentes adaptam um modelo prescritivo (genérico) a suas necessidades.

CICLO DE VIDA

- Independente do modelo de ciclo de vida escolhido algumas atividades sempre devem aparecer:
  - atividades de **definição** do problema (**o que**);
  - de **construção** de uma solução (**como**);
  - de **manutenção** do sistema depois da sua entrega ao cliente.

CICLO DE VIDA

Em geral, os ciclos de vida envolvem as seguintes **fases**:

– ***Planejamento***
– ***Análise e Especificação de Requisitos***
– ***Projeto***
– ***Implementação***
– ***Testes***
– ***Entrega e Implantação***
– ***Operação***
– ***Manutenção***

CICLO DE VIDA

- ***Planejamento***

– Fornece uma estrutura que possibilita ao gerente fazer **estimativa** iniciais de **recursos**, **custos** e **prazos**;

– O **escopo** do software é estabelecido;

–Um **plano de projeto** deve ser elaborado **configurando o processo** a ser utilizado;

– Esta atividade faz parte da **gerência de projeto**.

CICLO DE VIDA

- ***Análise e Especificação de Requisitos***

– O **escopo** do software é **refinado**;

– Descreve **"o que"** o software deve fazer;

– Devem ser analisados o **domínio do problema** e o **domínio da solução**.

- ***Projeto***

– Utiliza a fase anterior como insumo;

– Envolve duas grandes etapas: projeto da **arquitetura do software** e projeto **detalhado**.

CICLO DE VIDA

- ***Implementação***

– O projeto é traduzido para uma forma passível de **execução pela máquina**.

- ***Teste***

–Testes de unidade e documentação dos resultados;

– Integração dos componentes e teste do software como um todo;

– Alguns modelos de processo preveêm a realização de testes já nas primeiras etapas.

CICLO DE VIDA

- ***Entrega e Implantação***

– O software deve ser **instalado** em **ambiente produção**.

– Envolve

- **Treinamento** de usuários;
- **Configuração** do ambiente de produção;
- **Conversão** bases de dados (se necessário).

– Principal propósito desta fase:

- Realiza-se os Testes de Aceitação (estabelecer que o software satisfaz os requisitos dos usuários).

CICLO DE VIDA

- ***Operação***
  – Após o teste de aceitação, o software passa a ser **utilizado de fato** em **ambiente de produção**.
- ***Manutenção***

– **Adaptativas**

– **Corretivas**

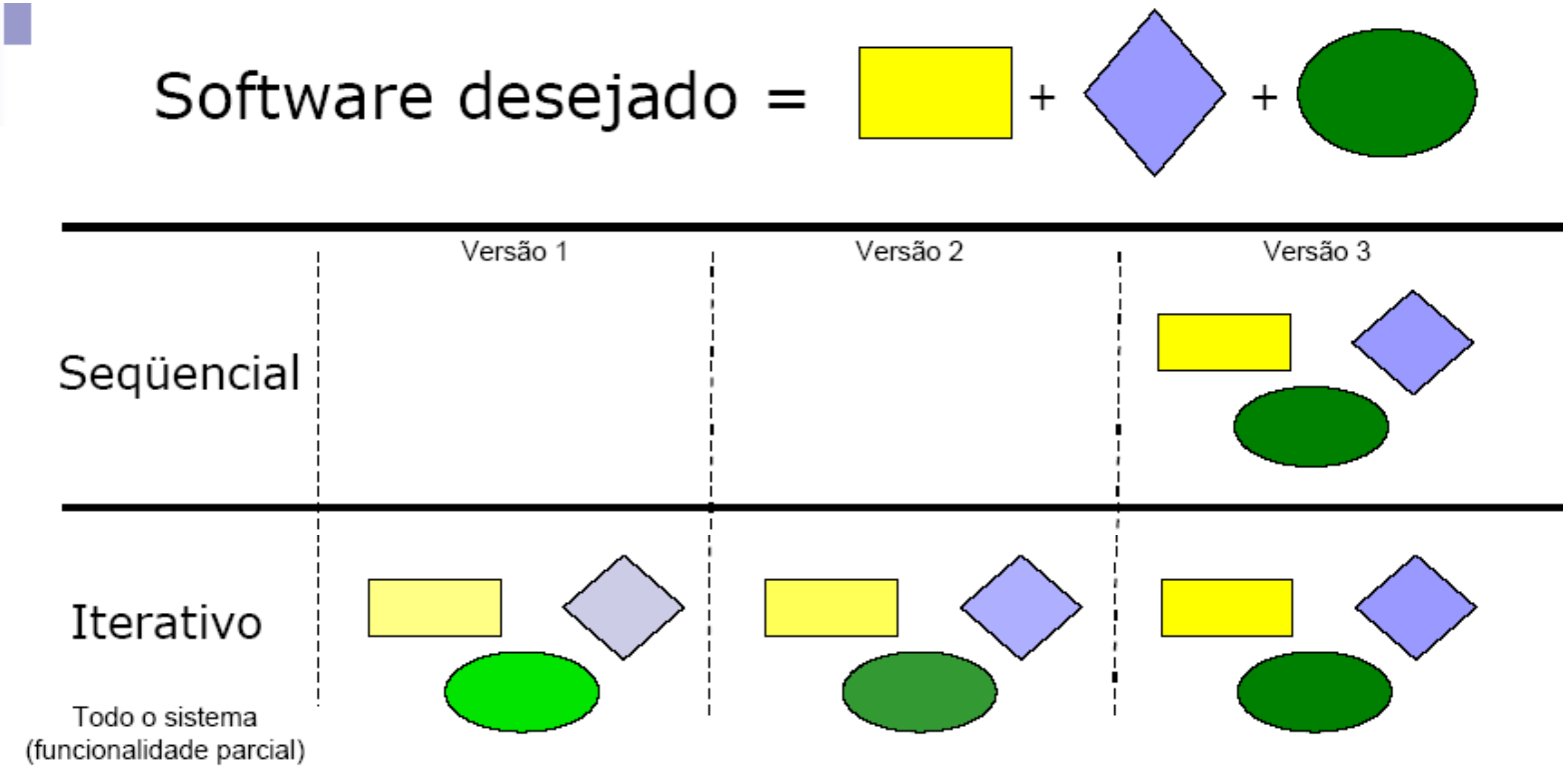
– **Evolutivas**

CICLO DE VIDA

- ***Modelos de Ciclo de Vida***

– **Sequenciais**
– **Iterativos/Incrementais**
– **Híbridos**

CICLO DE VIDA



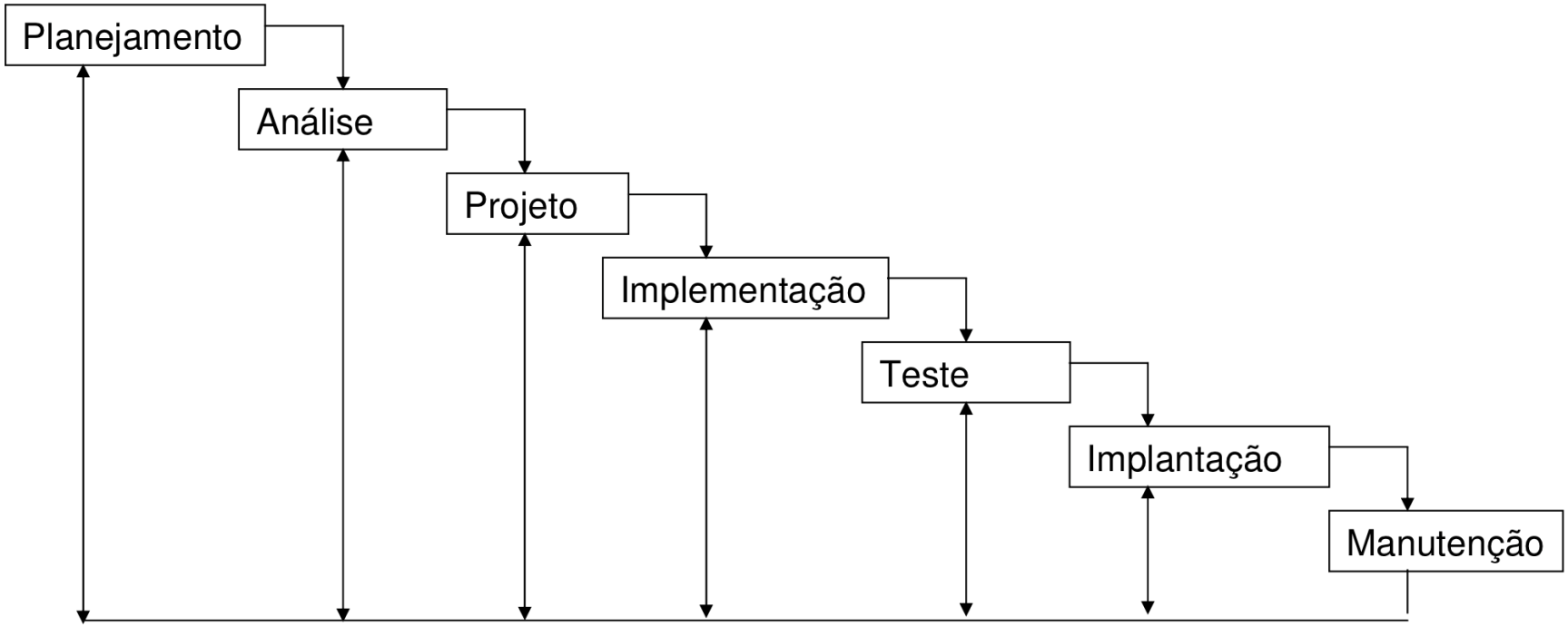
MODELO SEQUENCIAL

MODELO SEQUENCIAL - CASCATA

- Clássico, requer abordagem sistemática e sequencial ao desenvolvimento de software

- Principal característica
  - "**O resultado de uma fase é a entrada da próxima**"

MODELO SEQUENCIAL - CASCATA



MODELO SEQUENCIAL - CASCATA

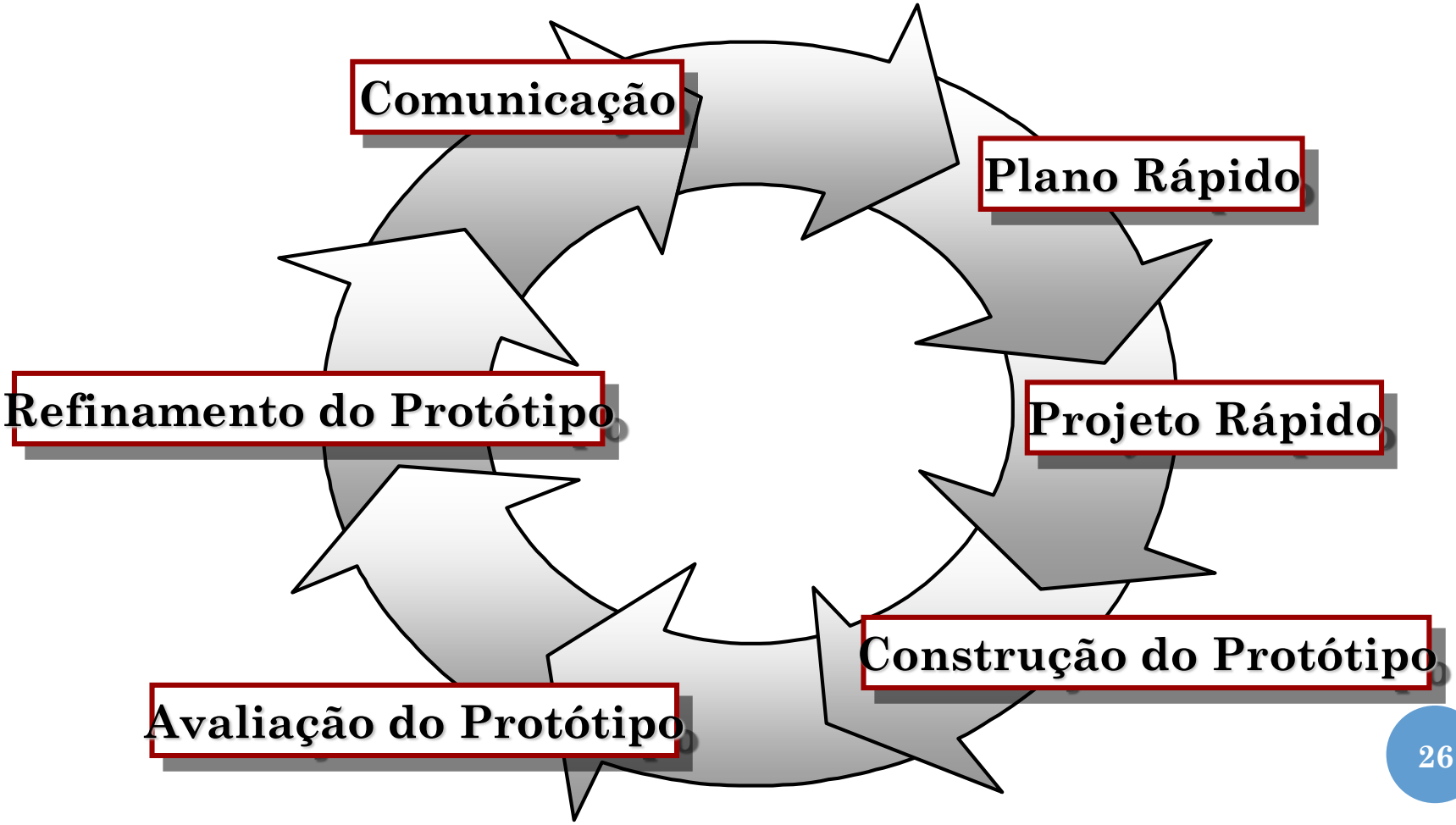
- **Problemas**
  - Em projetos reais, é difícil estabelecer todos os requisitos no início de um processo (incertezas)
  - Difícil acomodar mudanças com o processo em andamento, pois uma fase deve estar completa para passar para a próxima (sem paralelismo)
    - Inflexibilidade em estágios distintos dificulta resposta aos requisitos de mudança do cliente
  - Cliente paciente
    - Uma versão executável só fica disponível em uma etapa avançada do desenvolvimento

MODELO ITERATIVO/INCREMENTAL

MODELO ITERATIVO

- Modelos de processo iterativos que permitem desenvolver versões cada vez mais completas de um software
  - Desenvolve-se uma implementação inicial, expondo-a aos comentários do usuário
  - Depois, refina-se esse resultado por meio de várias versões até que seja desenvolvido um sistema adequado

MODELO ITERATIVO - **PROTOTIPAGEM**



MODELO ITERATIVO - **PROTOTIPAGEM**

- **Vantagens**
  - Usuários têm o "sabor" de um sistema real precocemente
  - Desenvolvedores conseguem "entender" o sistema e construir "algo" em prazo curto
- **Desvantagens**
  - Cliente "pensa" estar usando uma versão **operacional**
  - Concessões **equivocadas** do desenvolvedor para entregar logo o protótipo (Algoritmo ineficiente, SO ou linguagem inapropriados)
  - O descartamento do protótipo pode ser visto com perda de tempo para o cliente

MODELO ITERATIVO - **PROTOTIPAGEM**

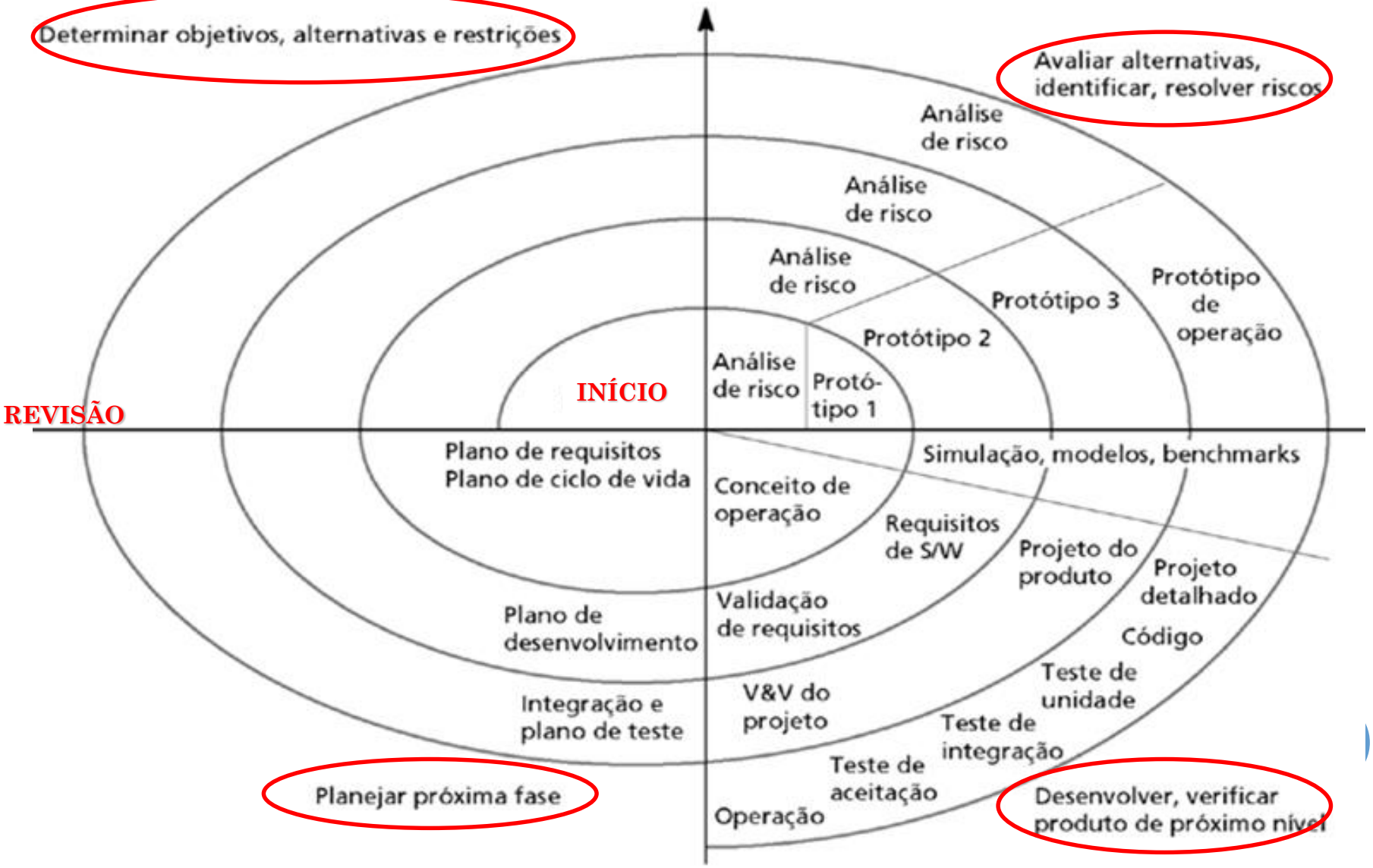
- **Problemas?**

- Falta de visibilidade de processo
  - Inviabilidade econômica de se produzir documentação para cada versão desenvolvida rapidamente
  - Falta de produtos regulares inviabilizam a medição do progresso do projeto
- Software freqüentemente mal estruturado
  - Mudanças contínuas tendem a corromper a estrutura do software, tornando-a cada vez mais onerosa e difícil
- Habilidades especiais podem ser solicitadas
  - Exemplos, linguagens para prototipação rápida (ex: Lua)

MODELO ITERATIVO - **ESPIRAL**

- Processo representado como uma espiral e cada volta na espiral representa uma iteração
  - Sem fases definidas (ex: comunicação, projeto) as voltas na espiral são escolhidas com base no que é requisitado
- Processo direcionado a riscos e planejamento
  - Riscos são explicitamente avaliados e resolvidos ao longo do processo

MODELO ITERATIVO - **ESPIRAL**



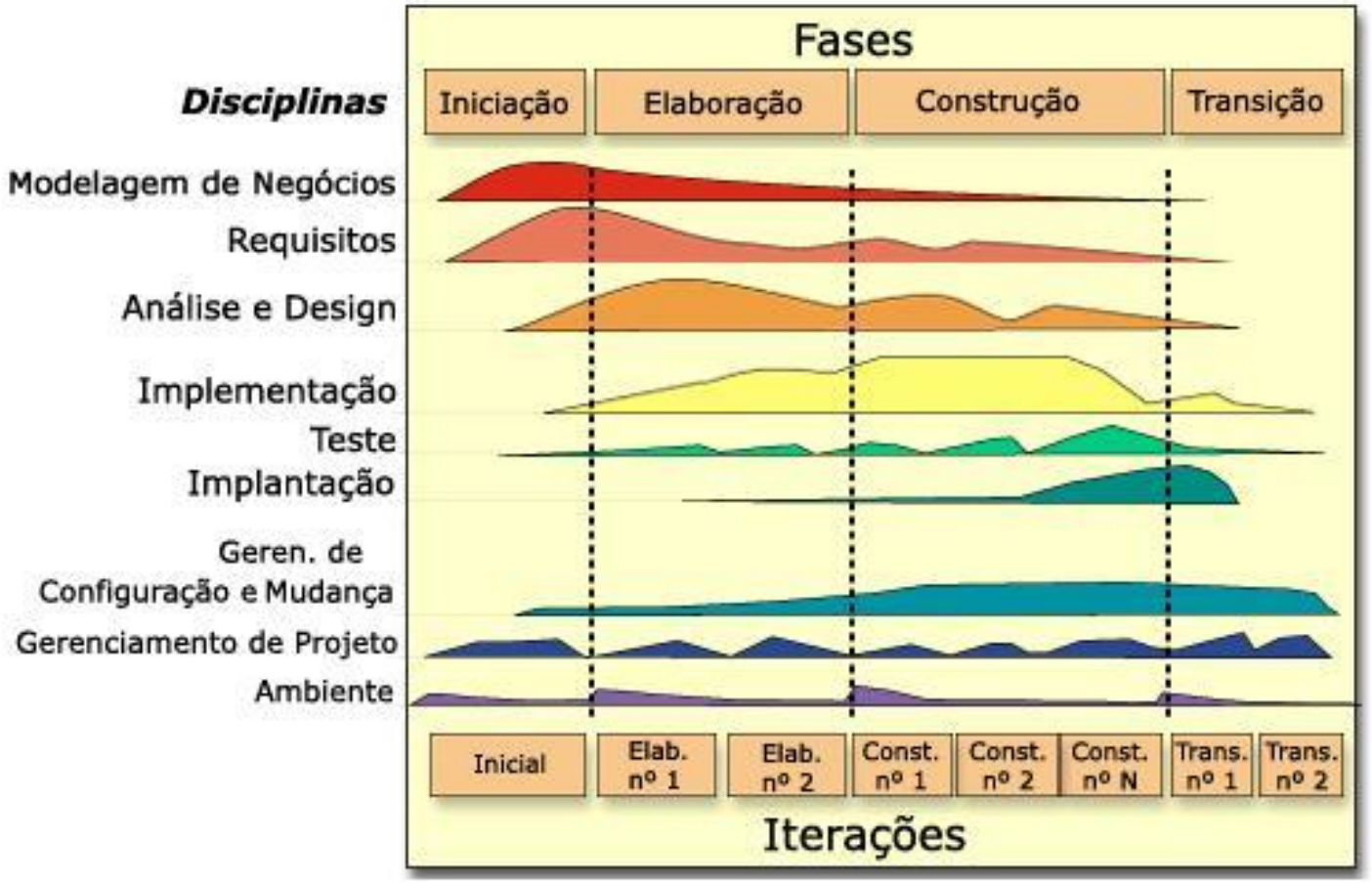
MODELO ITERATIVO - **ESPIRAL**

- **Vantagens?**
  - Riscos são gerenciados cedo e ao longo do processo – reatividade a riscos, que são reduzidos antes de se tornarem problemáticos
  - Usa prototipação para reduzir riscos
  - Software evolui enquanto o projeto prossegue – erros/alternativas não atrativas são eliminadas cedo
  - Planejamento é construído sobre o processo – cada ciclo inclui um passo de planejamento para auxiliar o monitoramento do projeto

MODELO HÍBRIDO

MODELO HÍBRIDO - **RUP**

- **Rational Unified Process (RUP)**



Principal inovação: separação das fases e disciplinas

DESENVOLVIMENTO ÁGIL

DESENVOLVIMENTO ÁGIL

- Manifesto para o Desenvolvimento Ágil
  - Assinado em **2001** por Kent Beck e outros 16 desenvolvedores, autores e consultores de software

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

- *Indivíduos e interações acima de processos e ferramentas*
- *Software operacional acima de documentação completa*
- *Colaboração dos clientes acima de negociação contratual*
- *Respostas a mudanças acima de seguir um plano"*

DESENVOLVIMENTO ÁGIL

- Surgiram de um esforço para sanar fraquezas reais e perceptíveis da engenharia de software convencional
- Oferece benefícios importantes

MAS, não é indicado para todos os tipos de projetos, produtos, pessoas e situações

DESENVOLVIMENTO ÁGIL

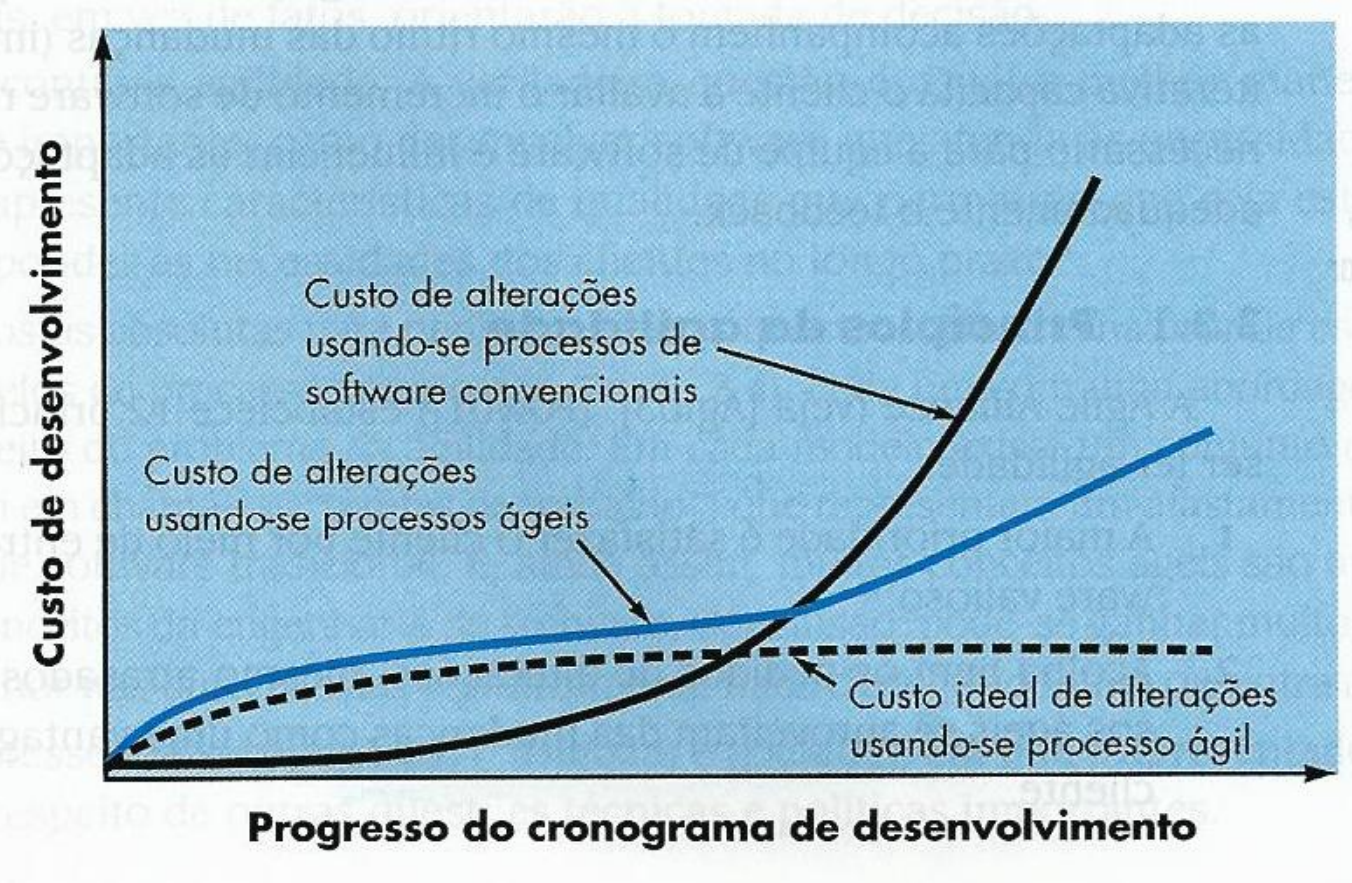
DESENVOLVIMENTO ÁGIL

- **AGILIDADE**
  - Entregar versões funcionais em prazos curtos
  - Estar preparado para requisitos mutantes
  - Pessoal de negócios e desenvolvedores juntos
  - Cliente é considerado parte da equipe (visão nós e eles)
  - Troca de informações através de conversas diretas
  - Plano de projeto deve ser flexível

DESENVOLVIMENTO ÁGIL

- **Agilidade e Custo das Mudanças**
  - Desenvolvimento de software convencional afirma que os custos com de mudanças aumentam de forma não linear conforme o projeto avança
  - Defensores da agilidade argumentam que o processo ágil bem elaborado "achata" o custo da curva de mudança
    - Processo ágil envolve entregas incrementais
    - Custo das mudanças é atenuado com entrega incremental associada a outras práticas ágeis: testes contínuos de unidade e programação por pares

DESENVOLVIMENTO ÁGIL



DESENVOLVIMENTO ÁGIL

Processo Ágil?

- Processo capaz de administrar a *imprevisibilidade*
  - Processo facilmente adaptável
  - Adaptar incrementalmente
  - Equipe precisa de feedback do cliente para adaptações incrementais
  - Catalisador para feedback do cliente é um protótipo operacional ou parte de um sistema operacional entregues em curtos períodos de tempo

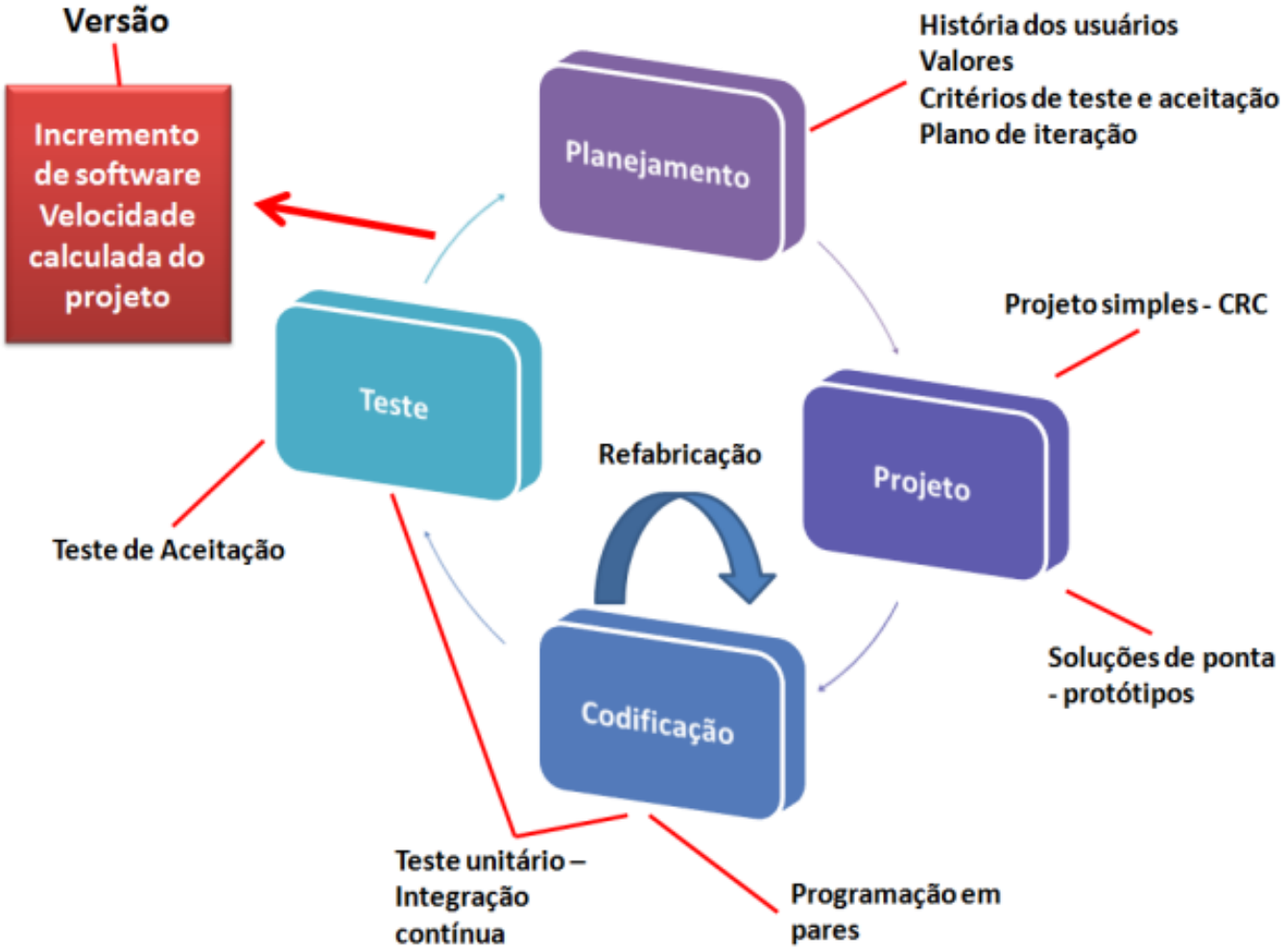
DESENVOLVIMENTO ÁGIL

Abordagens?

- **Extreme Programming – XP – Programação Extrema**
- **Scrum**
- Industrial XP – IXP – Programação extrema industrial
- Adaptive Software Development – ASD – Desenvolvimento de software adaptativo
- Dynamic Systems Development Method – DSDM – Método de Desenvolvimento de Sistemas Dinâmicos
- Crystal

DESENVOLVIMENTO ÁGIL

XP – Extreme Programming

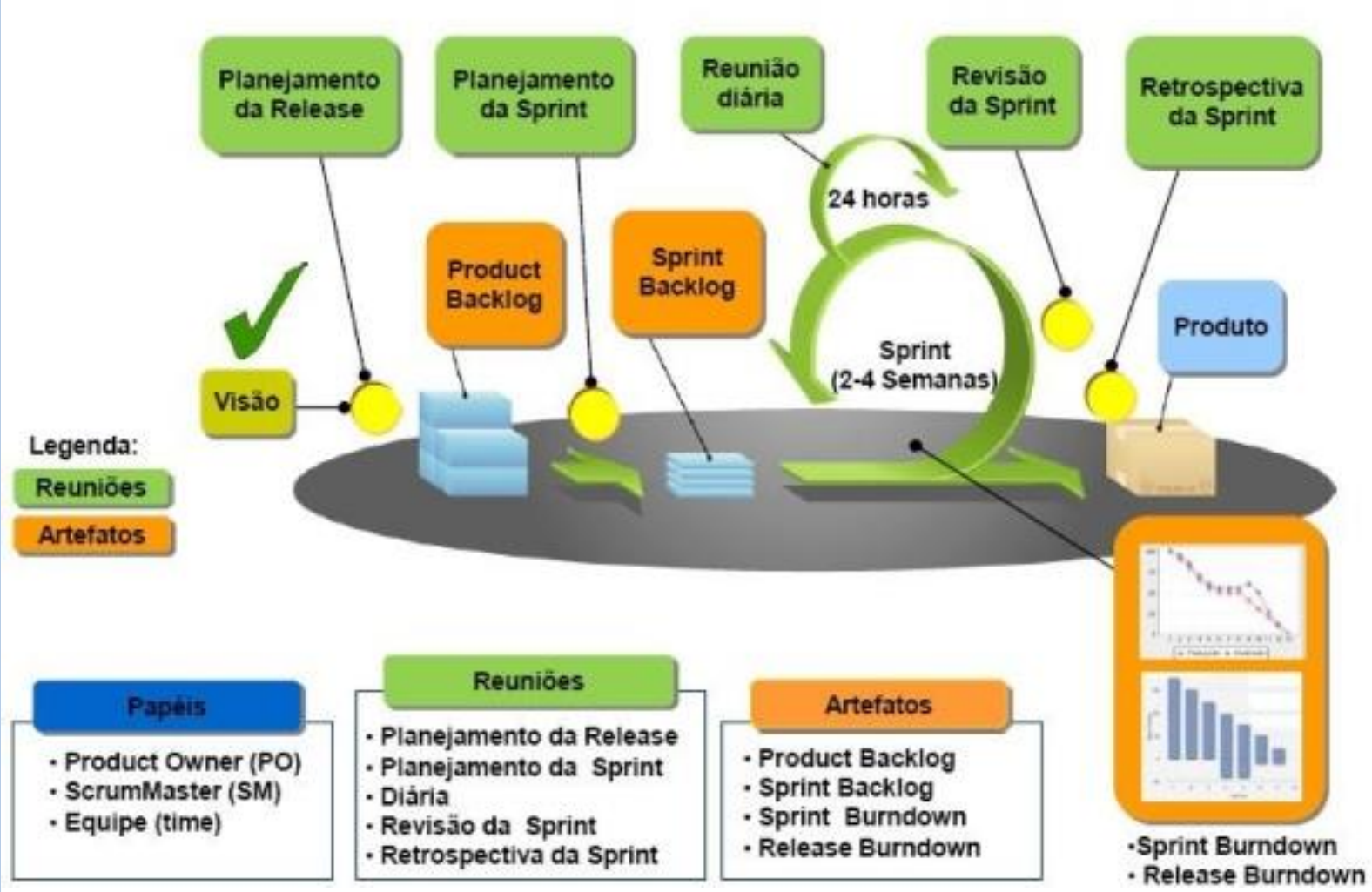


DESENVOLVIMENTO ÁGIL

Scrum

- Enfatiza o uso de padrões de processos de software
  - ***Backlog***: lista com requisitos pendentes com prioridades
  - ***Sprints***: unidades de trabalho com requisitos estabelecidos e prazo de entrega (geralmente 2 à 4 semanas)
  - ***Alterações***: não são introduzidas em Sprints já iniciados – membros trabalham em ambiente de curto prazo, mas estável
  - ***Reuniões Scrum***: reuniões curtas (15 minutos) realizadas diariamente
    - O que realizou desde a última reunião?
    - Quais obstáculos está encontrando?
    - O que planeja realizar até a próxima reunião?

DESENVOLVIMENTO ÁGIL



Dúvidas