INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
BAHIA

Modelos de Processo de *Software*

Profª Jocelma Rios

Jun/2013

O que pretendemos:

- Apresentar os conceitos básicos de processos de *software*
- Descrever os principais modelos de processos de *software*, elucidando suas vantagens e desvantagens
- Refletir sobre as vantagens e desvantagens de cada modelo para os mais variados tipos de *software* e contexto nos quais eles se aplicam

Processo de *Software*

Conjunto de atividades e resultados associados que levam à produção de um produto de *software*

→ Não há um processo ideal e até dentro da mesma empresa pode haver muitos processos diferentes utilizados para o desenvolvimento de *software*

Padrões de processo

Um padrão de processo descreve um problema de processo encontrado durante o trabalho de engenharia de *software*, identificando o ambiente onde foi encontrado e sugerindo uma ou mais soluções comprovadas para o problema.

Em termos mais genéricos, um padrão de processo fornece um modelo → um método consistente para descrever soluções de problemas no contexto do processo de *software*.

Modelos de Processo de *Software*

- São utilizados para explicar diferentes abordagens do desenvolvimento de *software*
- Definem a sequência em que as atividades do processo serão realizadas

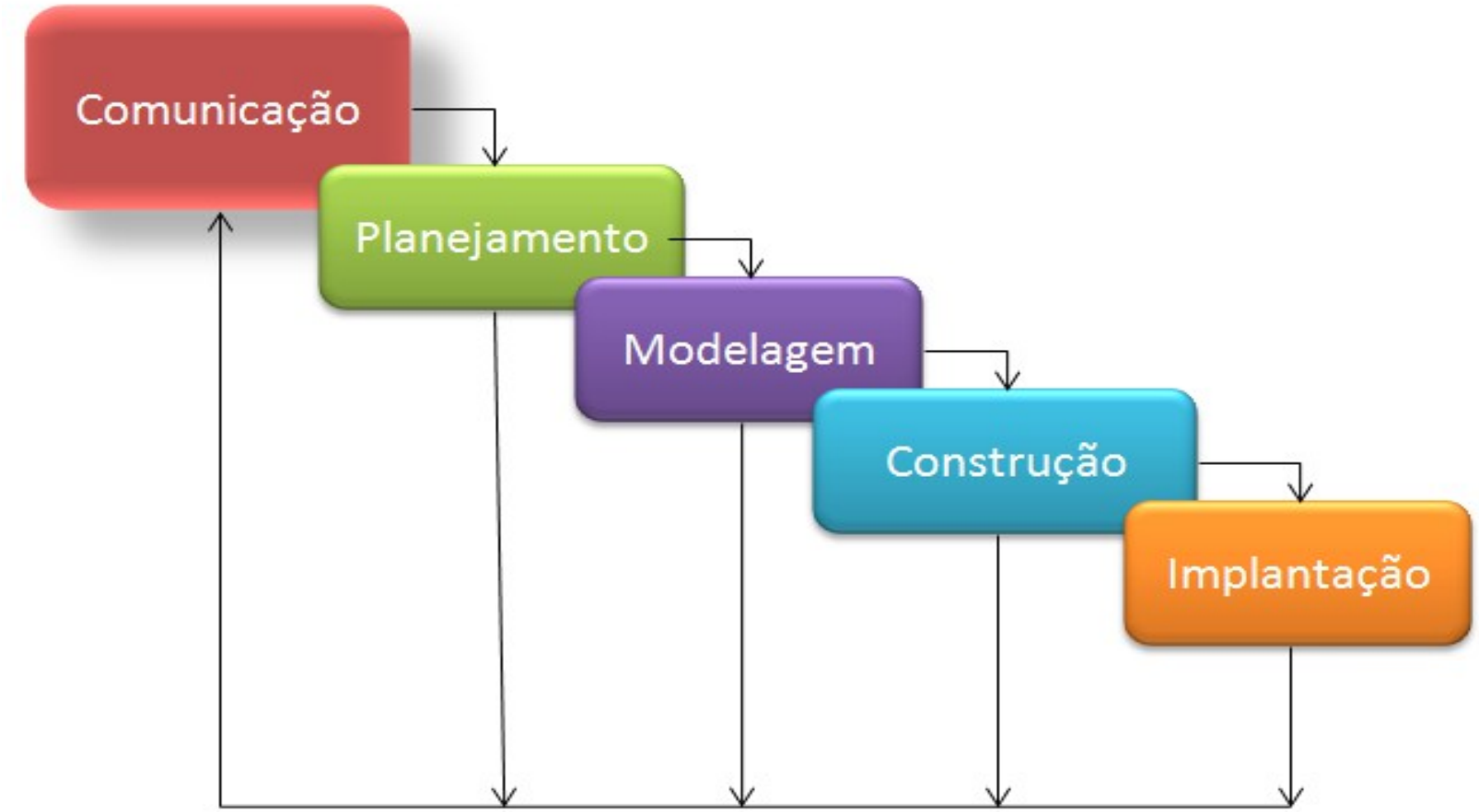
Exemplos de processo de *software*:

1. Modelo Cascata ou Clássico
2. Engenharia de *Software* Baseada em Componentes
3. Modelo Incremental
4. Modelo Evolucionário
5. Processo Unificado (será visto em detalhes adiante)

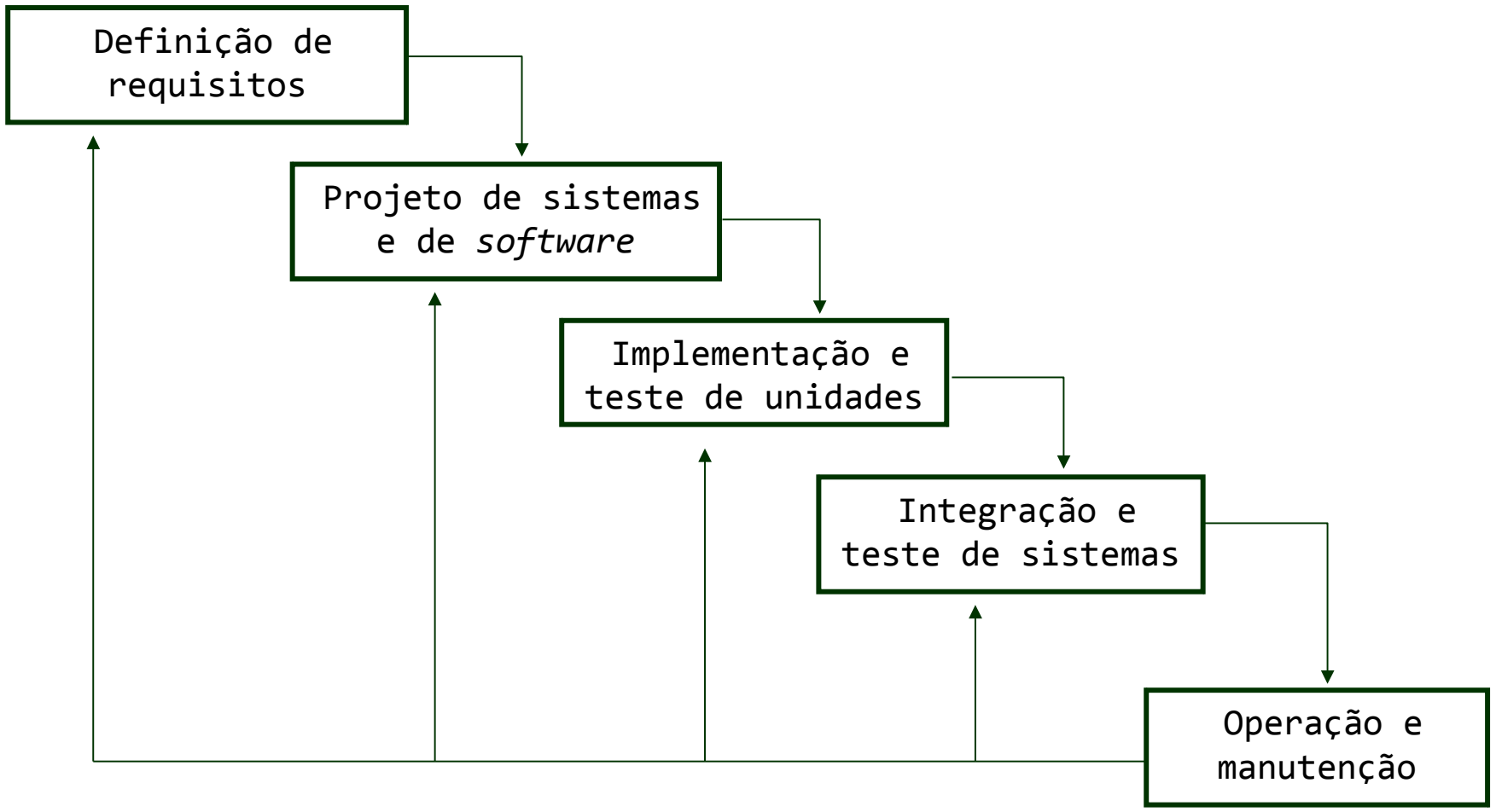
Modelo Cascata

- Primeiro modelo publicado do processo de desenvolvimento de *software*
- Também conhecido como Ciclo de Vida Clássico ou Modelo Clássico
- Considera as atividades de especificação, desenvolvimento, validação e evolução, que são fundamentais ao processo, e as representa como fases separadas do processo, como a especificação de requisitos, o projeto de software, os testes e assim por diante

Modelo Cascata



Modelo Cascata



Abordagem de Sommerville

Fases do Modelo Cascata

1. Análise e definição de requisitos
(especificação de requisitos)

→ As funções, as restrições e os objetivos do sistema são estabelecidos por meio da consulta aos usuários do sistema

→ Em seguida, são definidos em detalhes e servem como uma especificação do sistema

Fases do Modelo Cascata

2. Projeto de sistemas e de *software*

→ Agrupa os requisitos em sistemas de *hardware* ou de *software*

→ Estabelece uma arquitetura do sistema geral

3. Implementação e teste de unidades

→ Durante esse estágio, o projeto de *software* é compreendido como um conjunto de programas ou de unidades de programa

→ O teste de unidade envolve verificar que cada unidade atenda a sua especificação

Fases do Modelo Cascata

4. Integração e teste de sistemas

→ As unidades de programa ou programas individuais são integrados e testados como um sistema completo a fim de garantir que os requisitos de *software* foram atendidos

→ Depois dos testes, o sistema de *software* é entregue ao cliente

Fases do Modelo Cascata

5. Operação e Manutenção

→ Normalmente, esta é a fase mais longa do ciclo de vida

→ O sistema é instalado e colocado em operação

→ A manutenção envolve corrigir erros que não foram descobertos em estágios anteriores do ciclo de vida ou aumentar as funções desse sistema à medida que novos requisitos são descobertos

Modelo Cascata

- Em princípio, o resultado de cada fase envolve um ou mais documentos que são aprovados

  → **Gera muita documentação, nem sempre utilizada posteriormente**

- A fase seguinte não deve iniciar até que a fase precedente tenha sido concluída

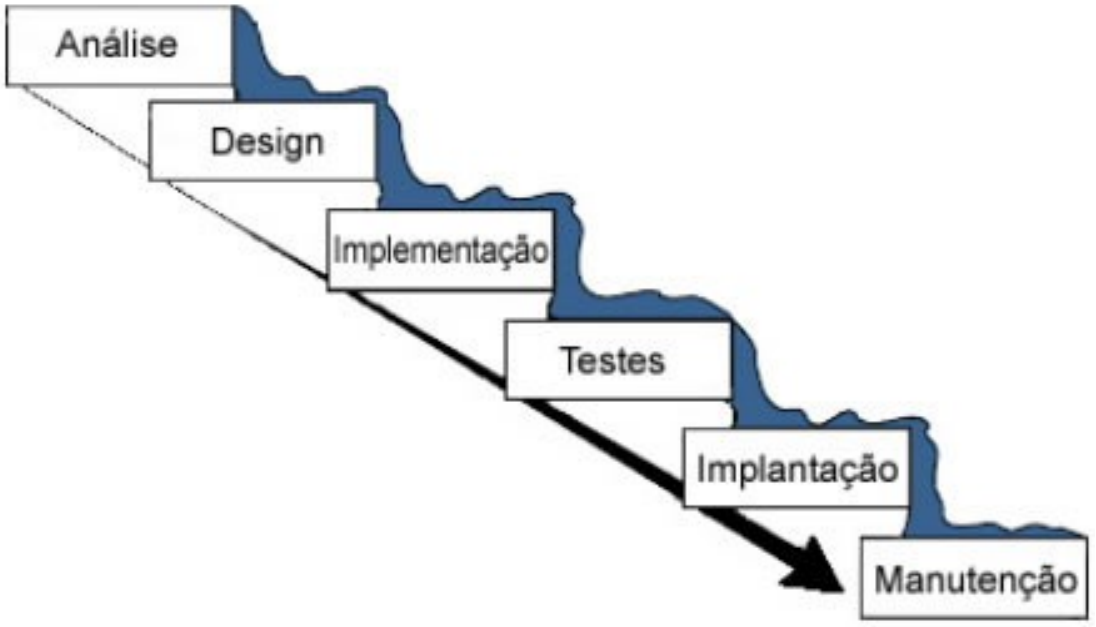
- O processo de *software* não é um modelo linear simples, pois envolve uma sequência de iterações das atividades

  → **problema do modelo cascata**

  → **inflexível divisão do projeto nesses estágios distintos**

Modelo Cascata

- O modelo cascata somente deve ser utilizado quando os requisitos forem bem compreendidos,
  - Contudo, ele reflete a prática da engenharia
  - Consequentemente, os processos de *software* com base nessa abordagem ainda são muito utilizados

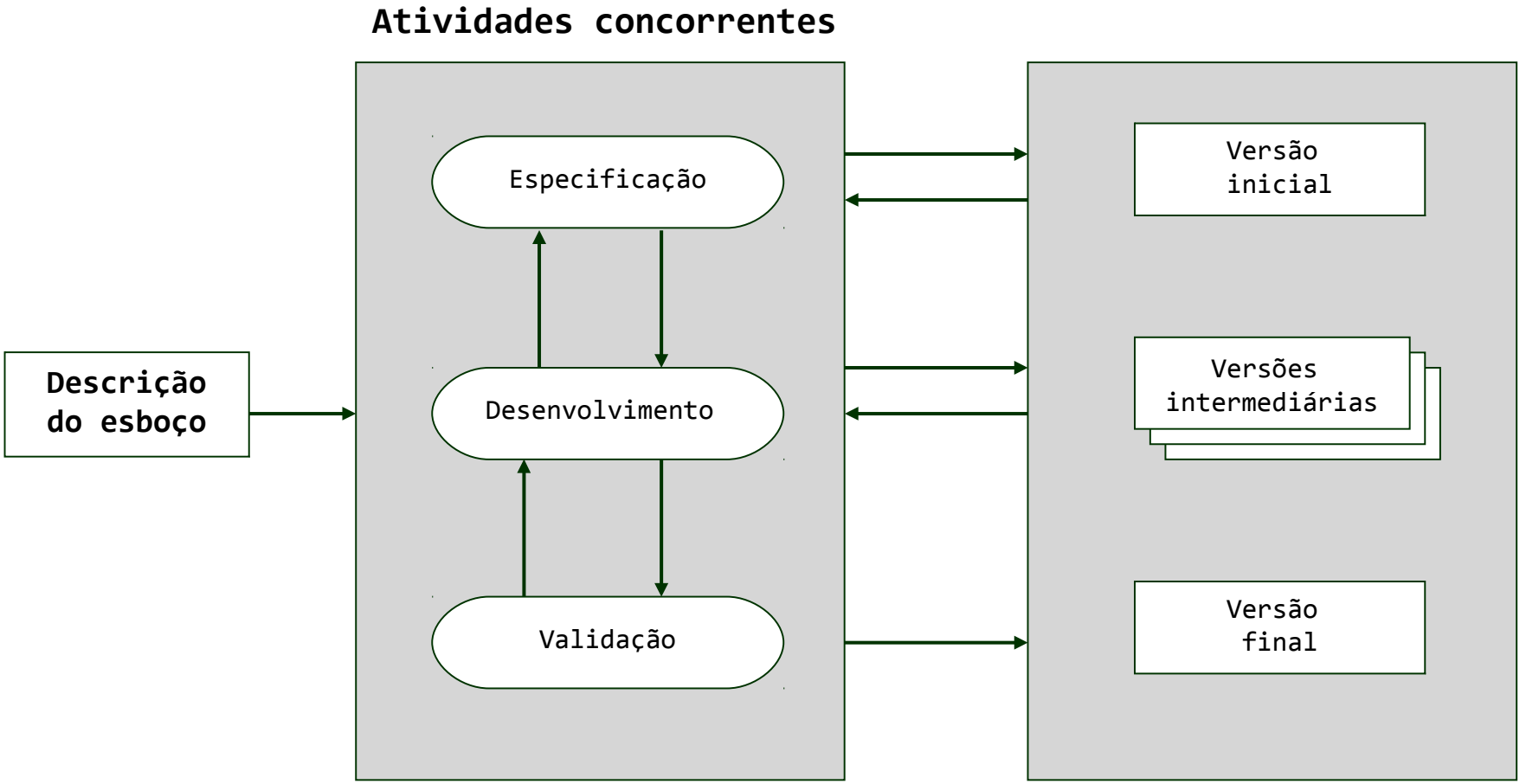


Modelo Evolucionário

- Tem como base a ideia de desenvolver uma implementação inicial, expor o resultado ao comentário do usuário e fazer seu **aprimoramento por meio de muitas versões**, até que um sistema adequado tenha sido desenvolvido

- Em vez de ter as atividades de especificação, desenvolvimento e validação em separado, todo esse **trabalho é realizado concorrentemente** com um rápido *feedback* por meio dessas atividades

Modelo Evolucionário



Modelo Evolucionário

Vantagens:

* A abordagem evolucionária do desenvolvimento de *software*, muitas vezes, é mais eficaz do que a abordagem em cascata, no sentido de produzir sistemas que atendam às necessidades imediatas dos clientes

* A especificação pode ser desenvolvida gradativamente. À medida que os usuários desenvolvem uma compreensão melhor de seus problemas, isso pode ser refletido na melhoria do *software* em construção

Modelo Evolucionário

Problemas:

* Como os *softwares* são desenvolvidos rapidamente, não é viável produzir documentos que reflitam cada versão do sistema

* Os *softwares* frequentemente são mal estruturados → a mudança constante tende a corromper a estrutura do software

* Incorporar modificações torna-se cada vez mais difícil e oneroso

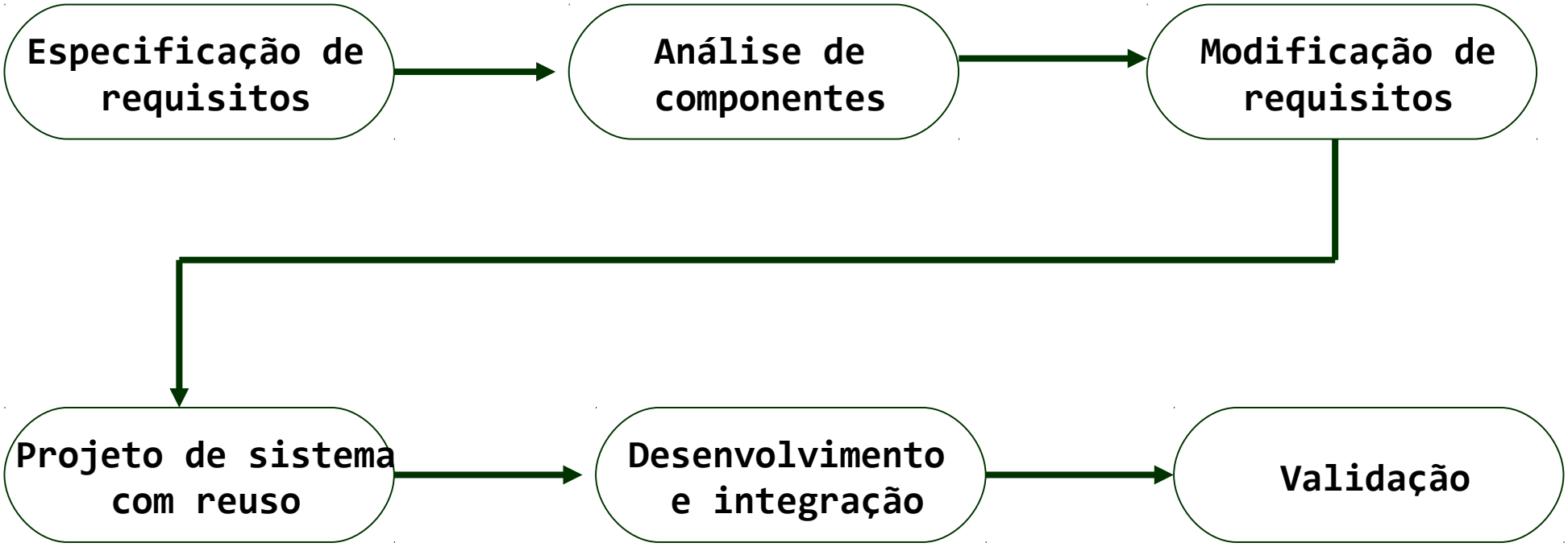
Engenharia de *Software* Baseada em Componentes

- Essa abordagem tem como base a existência de um número significativo de componentes reutilizáveis

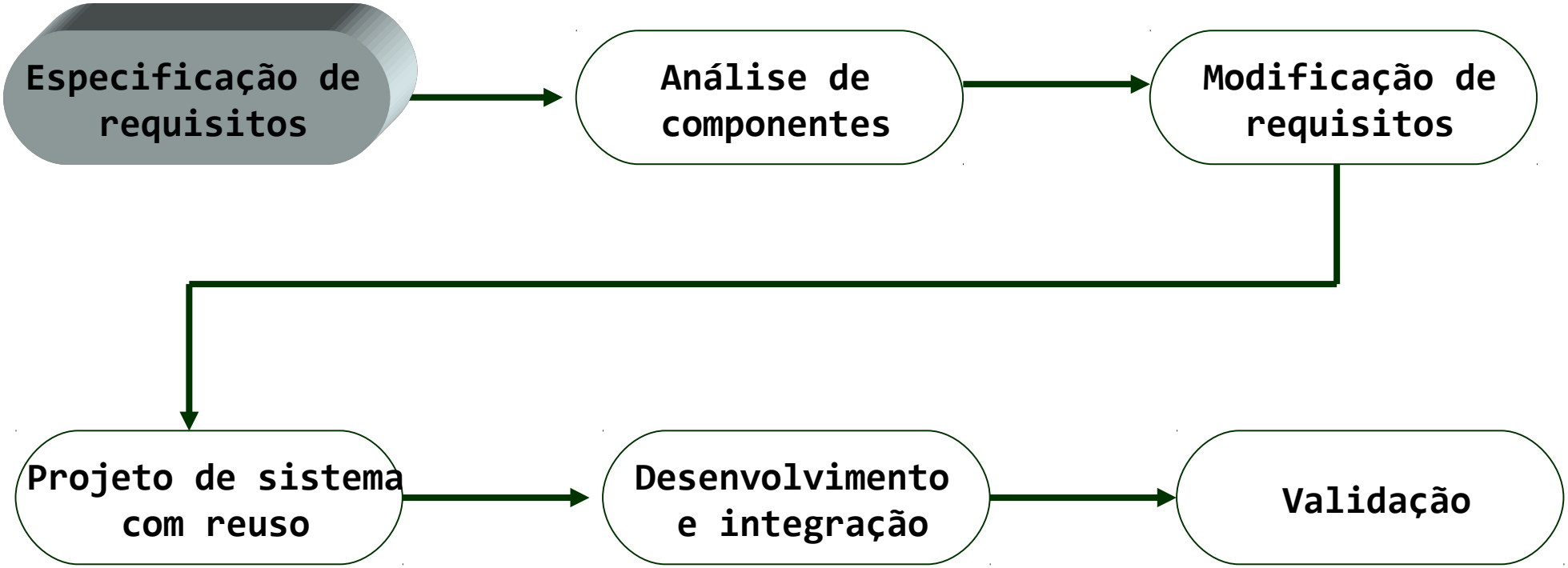
- O processo de desenvolvimento de *software* se concentra na integração desses componentes, ao invés de proceder ao desenvolvimento a partir do zero

Engenharia de *Software* Baseada em Componentes

Modelo genérico de processo para a engenharia de *software* baseada em componentes



Engenharia de Software Baseada em Componentes



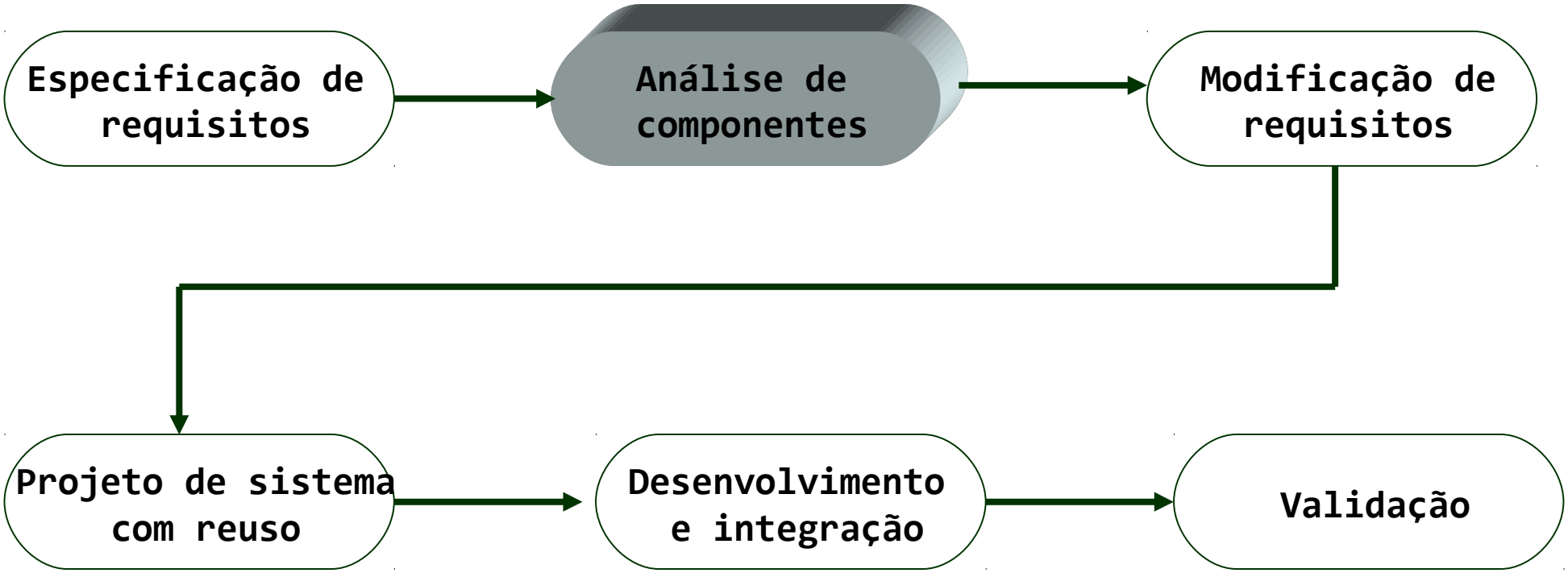
Engenharia de *Software* Baseada em Componentes

- **Especificação de Requisitos**
  → comparável com outros processos, como por exemplo, o modelo cascata

- As funções, as restrições e os objetivos do *software* são estabelecidos por meio da consulta aos usuários

- Em seguida, são definidos em detalhes e servem como uma especificação do *software*

Engenharia de *Software* Baseada em Componentes

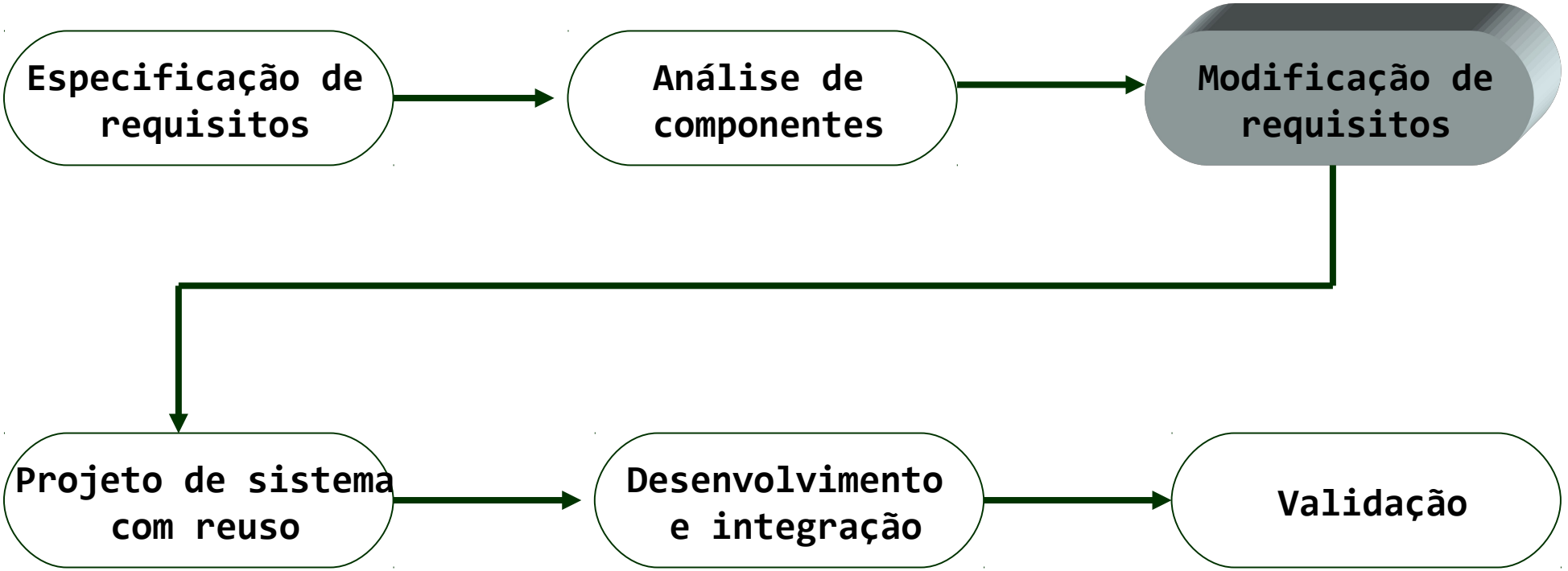


Engenharia de *Software* Baseada em Componentes

- **Análise de Componentes** → com base na especificação de requisitos, é feita uma busca de componentes para implementar essa especificação

- Nem sempre é possível encontrar uma combinação exata e os componentes que podem ser utilizados fornecem **somente** parte da funcionalidade requerida

Engenharia de *Software* Baseada em Componentes



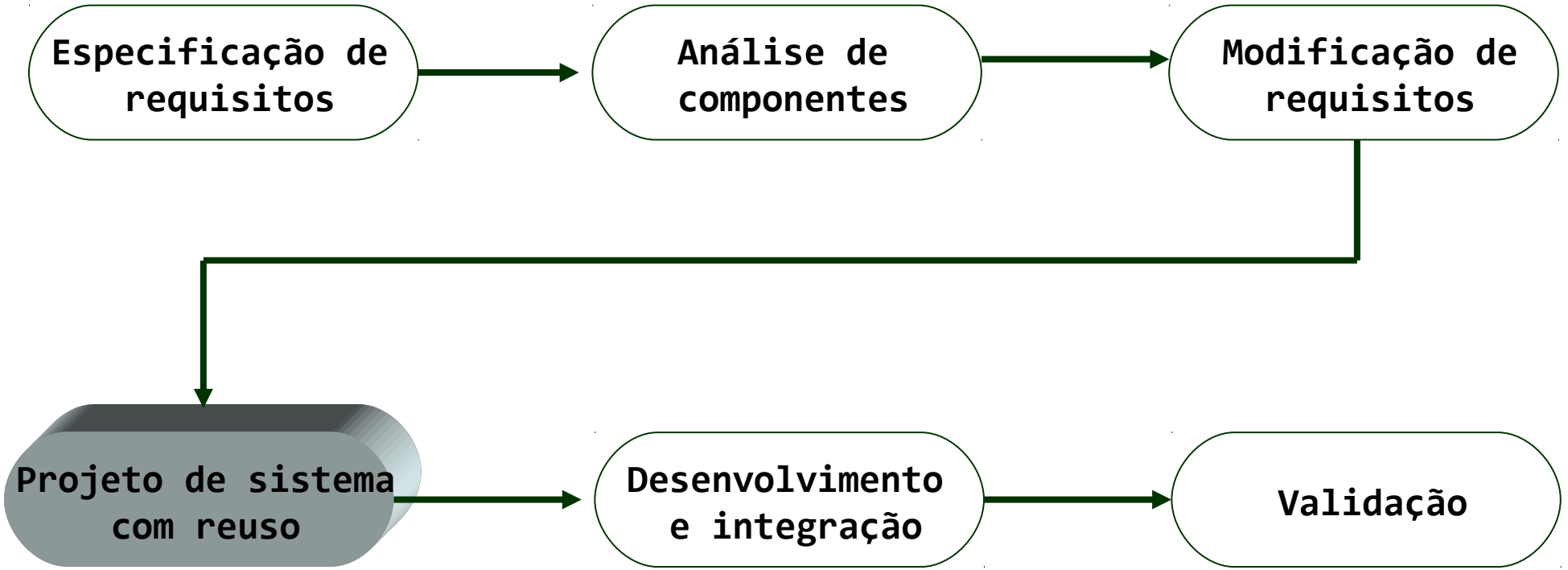
Engenharia de *Software* Baseada em Componentes

- **Modificação de requisitos** → durante esse estágio, os requisitos são analisados, utilizando-se as informações sobre os componentes que foram encontrados

- Eles são então modificados para refletir os componentes disponíveis

- Quando as modificações forem impossíveis, a atividade de análise de componentes é refeita, a fim de procurar soluções alternativas

Engenharia de *Software* Baseada em Componentes

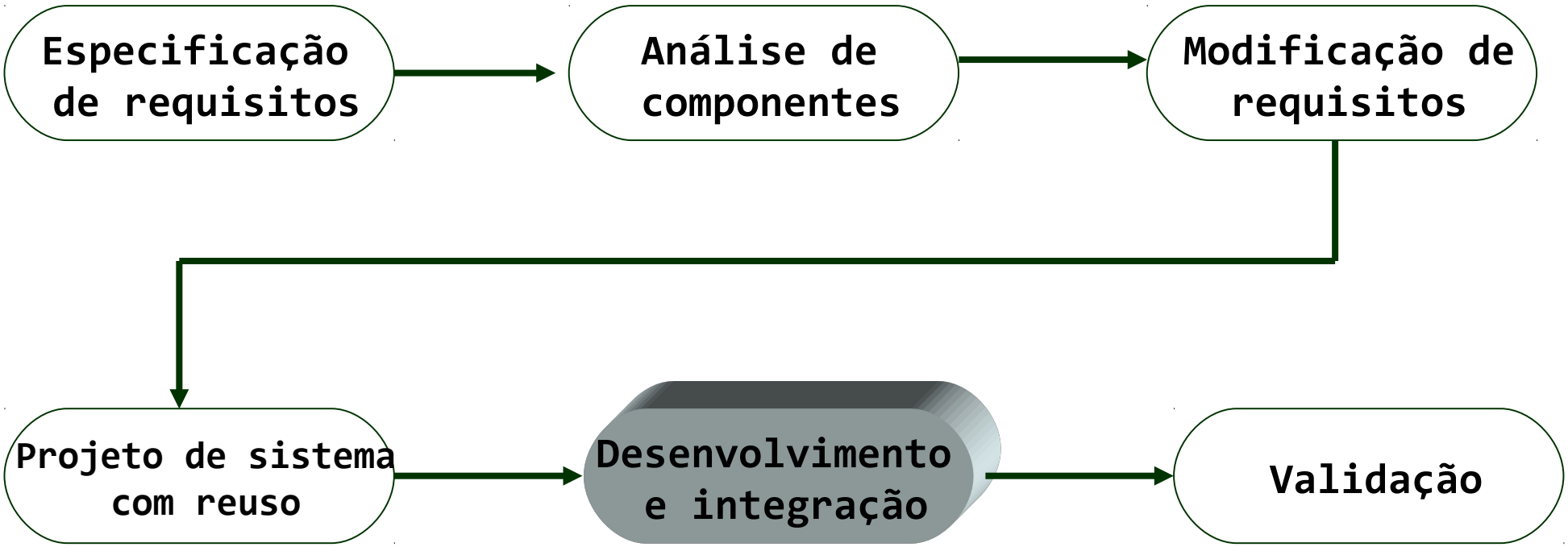


Engenharia de *Software* Baseada em Componentes

- **Projeto de sistema com reuso** → durante essa fase, a infraestrutura do sistema é projetada ou uma infraestrutura existente é reutilizada

- Os projetistas levam em conta os componentes que são reutilizados e organizam a infraestrutura para lidar com esse aspecto

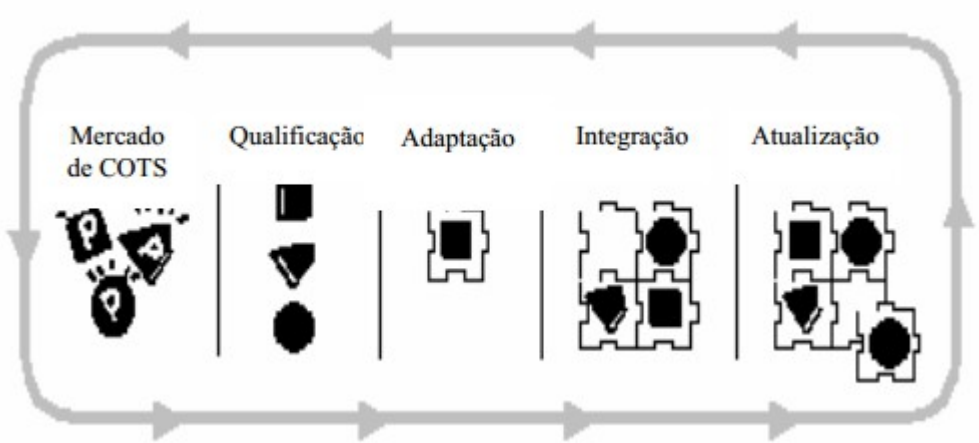
Engenharia de *Software* Baseada em Componentes



Engenharia de *Software* Baseada em Componentes

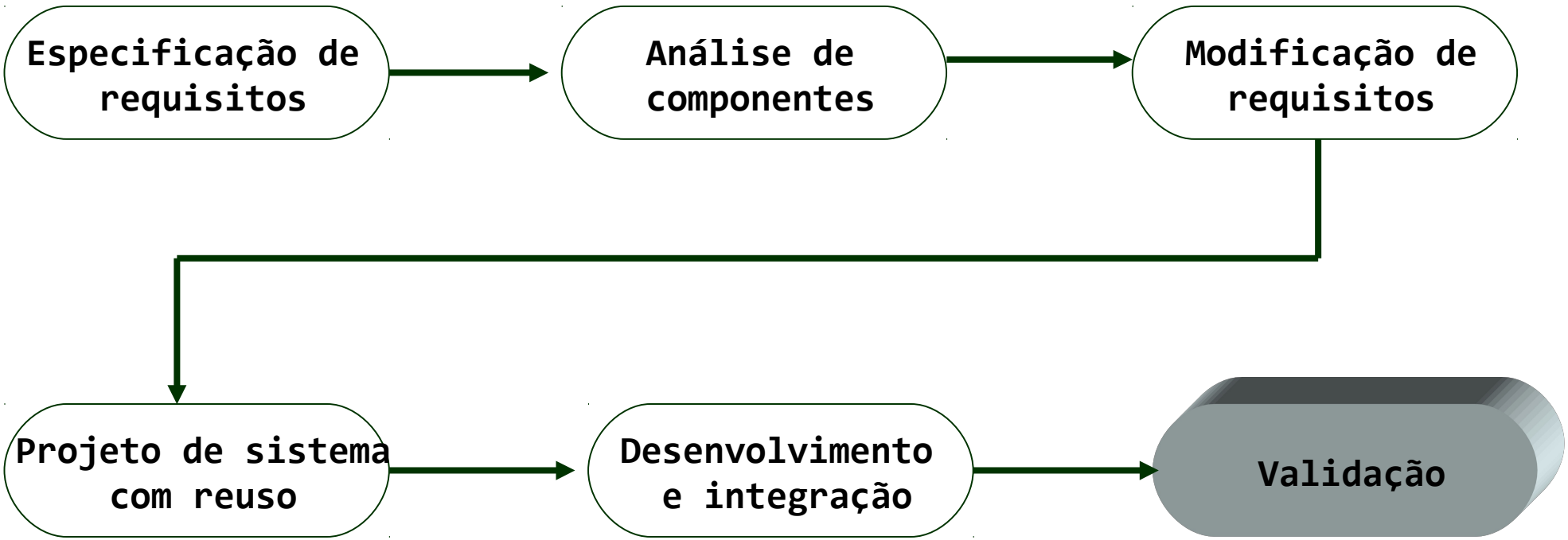
- **Desenvolvimento e integração**

  → o *software* que não puder ser comprado, será desenvolvido, e os componentes e sistemas COTS (*commercial off-the-shelf* – sistemas comerciais de prateleira) serão integrados, a fim de atender por completo a especificação do usuário



Saiba mais: http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER99/souza.pdf

Engenharia de *Software* Baseada em Componentes



Engenharia de *Software* Baseada em Componentes

- **Validação de sistema** → o *software* deve ser validado para garantir que atende a especificação do usuário

Engenharia de *Software* Baseada em Componentes

- **Vantagens**
  - Reduz a quantidade de *software* a ser desenvolvida, reduzindo custos e riscos
  - Geralmente propicia a entrega mais rápida do *software*

Engenharia de *Software* Baseada em Componentes

- **Problemas**
  - As adequações nos requisitos são inevitáveis, e isso pode resultar em um *software* que não atenda às reais necessidades dos usuários
  - O controle sobre a evolução do *software* se perde, uma vez que novas versões dos componentes reutilizáveis não estão sob o controle da organização que utiliza esses componentes

Processos Iterativos

- Os modelos apresentados anteriormente apresentam vantagens e desvantagens

- Para o desenvolvimento de grandes sistemas, pode haver a necessidade de utilizar **diferentes abordagens para diferentes partes**, de maneira que um **modelo híbrido** tem de ser utilizado

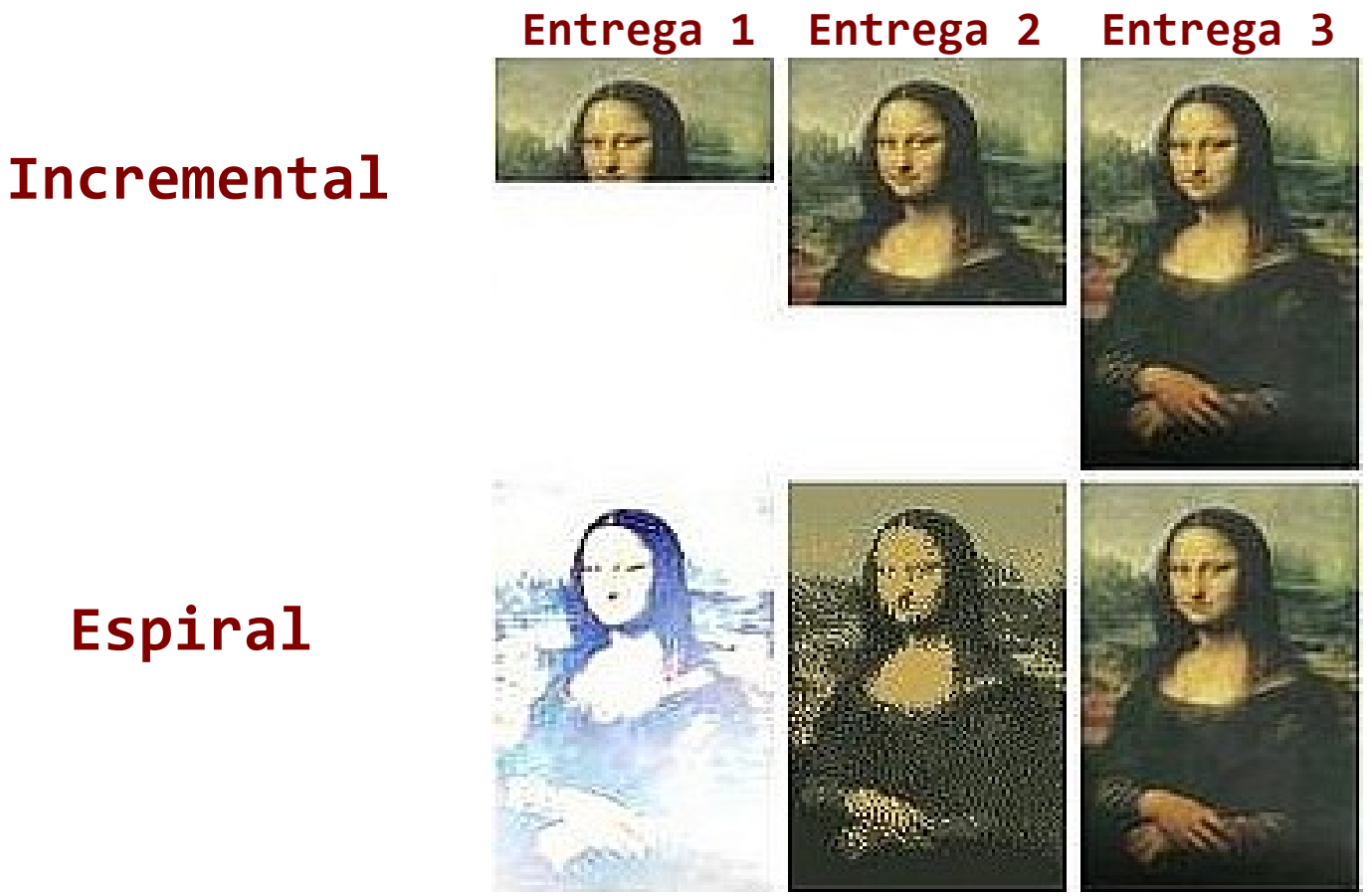
- Há também a necessidade de **iteração**, em que partes do processo são repetidas, à medida que os requisitos do *software* evoluem

Processos Iterativos

- Existem dois modelos híbridos, que apoiam diferentes abordagens do desenvolvimento e que foram explicitamente projetados para apoiarem a iteração do processo
  - Desenvolvimento Incremental
  - Desenvolvimento em Espiral

Processos Iterativos

Incremental vs. Espiral → uma metáfora



Modelo Incremental

- Possui uma abordagem mais simplista para a identificação dos requisitos, buscando obter um produto para entrega o mais rápido possível
- A cada entrega, os requisitos são refinados para que haja a expansão das funcionalidades
- Combina elementos dos fluxos de processos lineares e paralelos
  - Aplica sequências lineares, de forma escalonada, à medida que o tempo avança
  - 1º ciclo foca nos requisitos essenciais, mínimos para o funcionamento do *software*
  - Cada entrega gera um produto operacional, efetivamente

Proposta de Pressman

Modelo Incremental

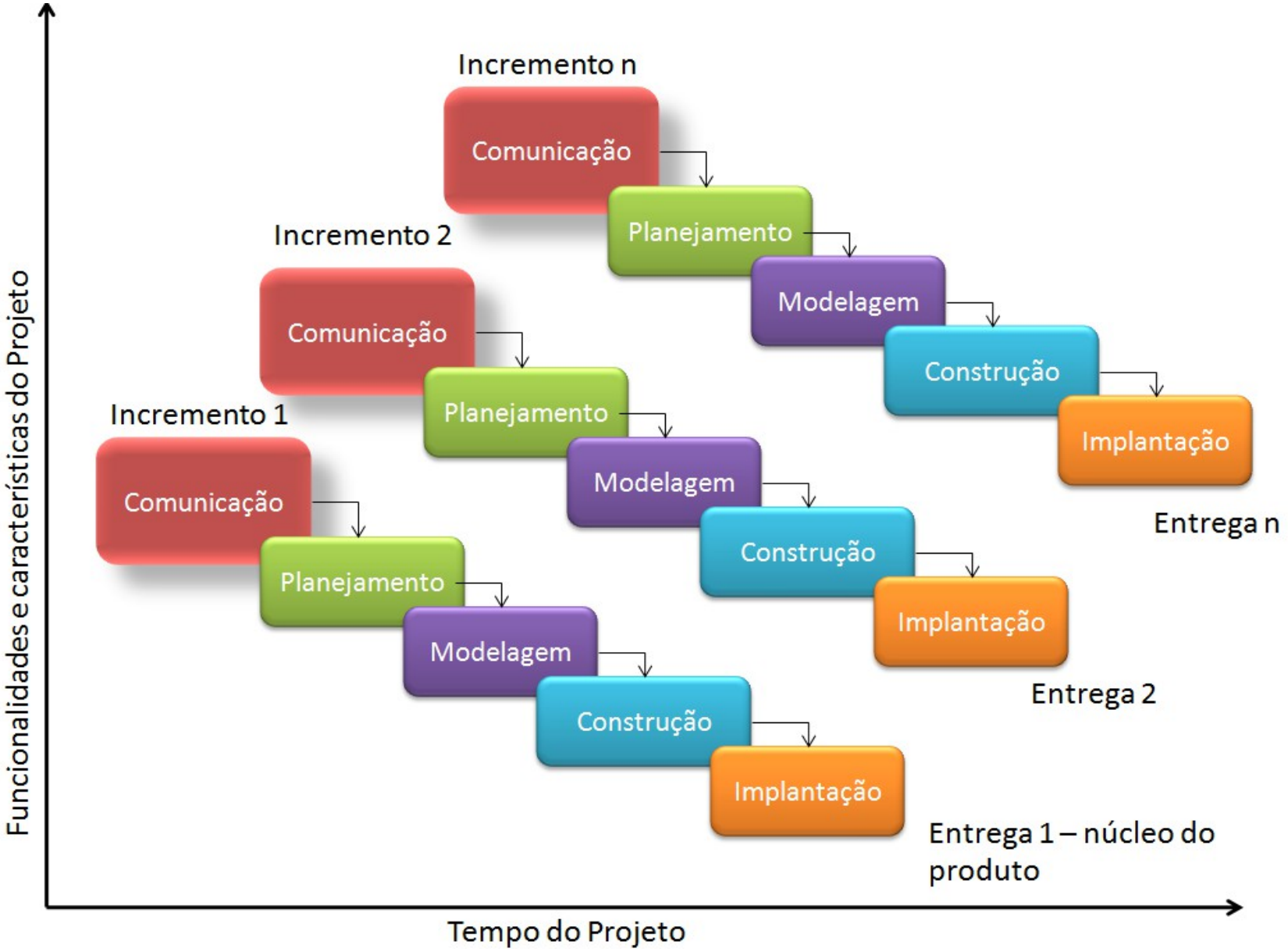
Situações em pode ser útil:

- Quando o *software* completo exige um *hardware* ainda não disponível e com data de entrega incerta
- Quando a equipe disponível é insuficiente para assumir o projeto completo, entregando num prazo factível para o usuário

Proposta de Pressman

Outro ponto de vista: http://blog.anascimento.net/category/gerenciamento

Modelo Incremental



Proposta de Pressman

Desenvolvimento Incremental

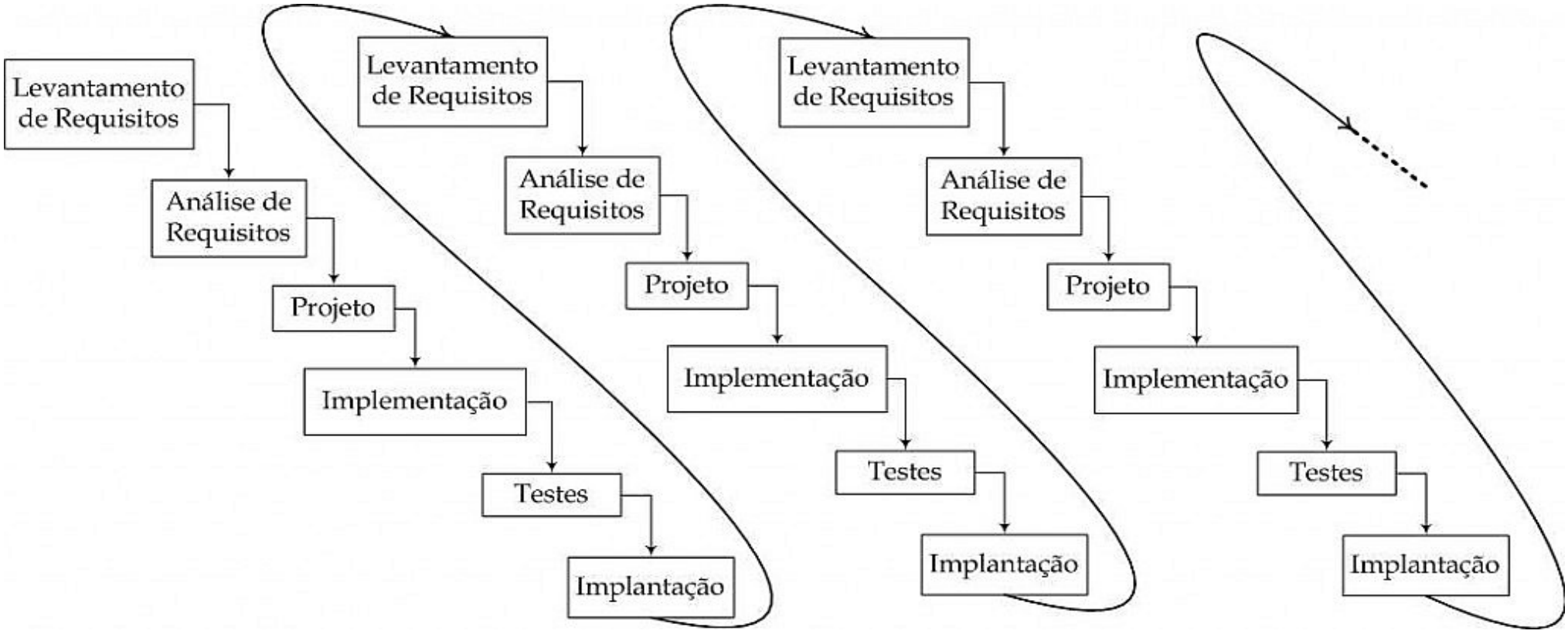
- É uma abordagem intermediária, que combina as vantagens do modelo cascata e do desenvolvimento evolucionário
- Em um processo de desenvolvimento incremental, os clientes identificam, em um esboço, as funções a serem fornecidas pelo sistema, definindo quais são **mais importantes** e quais são **menos importantes**
- Em seguida, é definida uma série de estágios de entrega, com cada um fornecendo um subconjunto das funcionalidades do *software*
- As funções prioritárias são entregues primeiramente ao usuário

Desenvolvimento Incremental

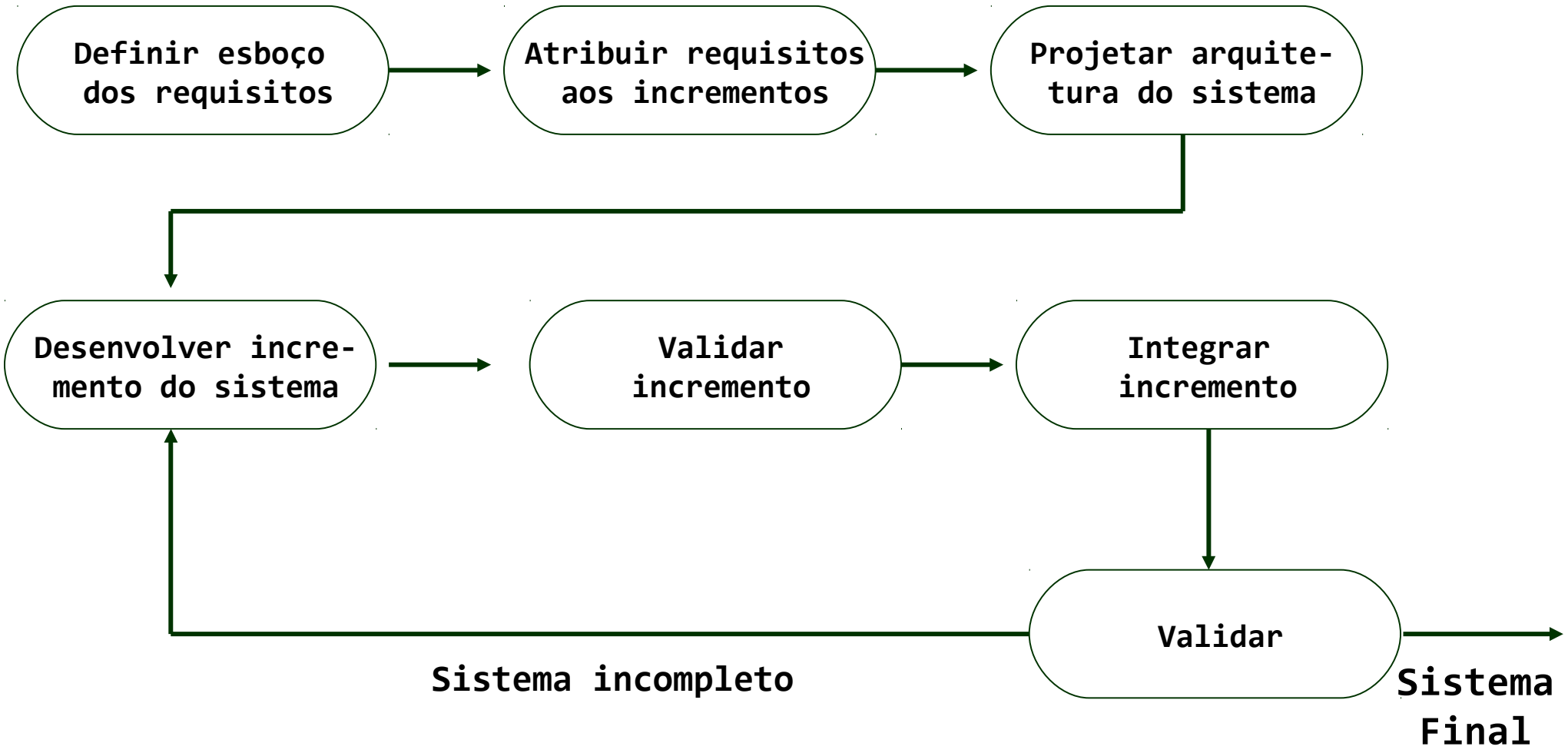
- Uma vez identificados os incrementos, os requisitos para as funções a serem entregues no primeiro incremento são definidos em detalhes
- Esse incremento é desenvolvido, utilizando-se o processo de desenvolvimento mais apropriado
- Uma vez que um incremento é concluído e entregue, os usuários podem colocá-lo em operação
- Isso significa que eles recebem com antecedência parte da funcionalidade do sistema, podendo experimentar o *software*, facilitando assim o esclarecimento dos requisitos para os incrementos seguintes

Desenvolvimento Incremental

- À medida que novos incrementos são concluídos, eles são integrados aos estágios existentes, melhorando a funcionalidade do *software* a cada novo estágio de entrega



Desenvolvimento Incremental



Desenvolvimento Incremental

- Não existe necessidade de utilizar o mesmo processo para o desenvolvimento de cada incremento

- Quando as funções têm especificação bem definida, o modelo de desenvolvimento em cascata pode ser utilizado

- Quando a especificação for mal definida, poderá ser utilizado um modelo de desenvolvimento evolucionário

Desenvolvimento Incremental

- **Vantagens**

  1. Os usuários não precisam esperar até que todo o *software* seja entregue, para então tirar proveito dele
  2. Existe um risco menor de fracasso completo do *software*. Embora possam ser encontrados problemas em alguns incrementos, é provável que alguns incrementos sejam entregues com sucesso
  3. Como as funções prioritárias são entregues primeiro, é inevitável que elas passem pela maior parte dos testes

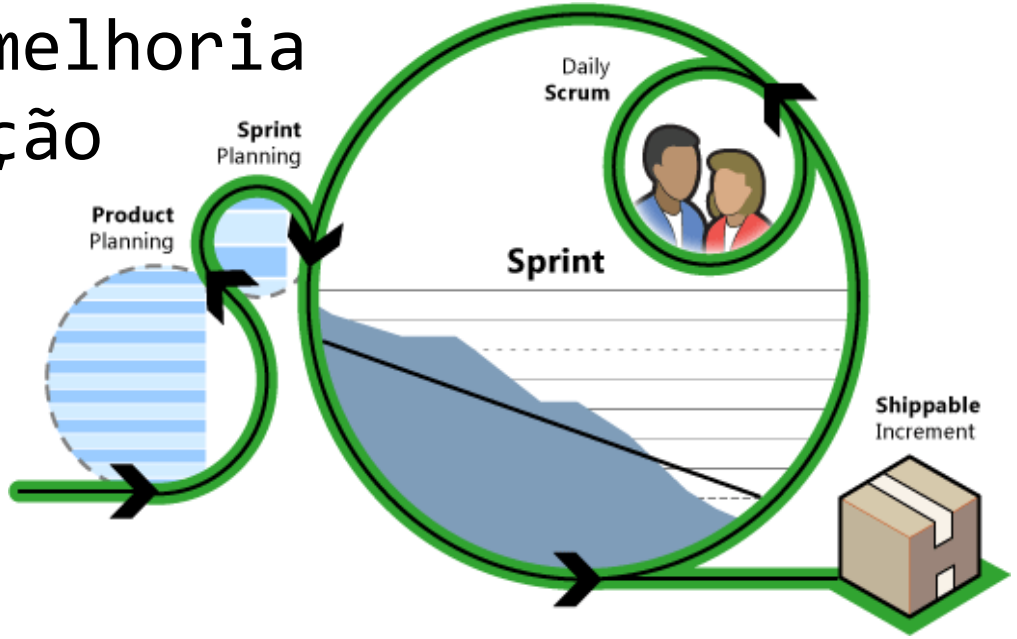
Desenvolvimento Incremental

- **Problemas**
  - Os incrementos devem ser relativamente pequenos, e cada incremento deve produzir alguma funcionalidade para o *software*
  - Pode, portanto, ser difícil mapear os requisitos dos usuários dentro de incrementos de tamanho correto

Desenvolvimento Incremental

- ***Extreme Programming*** **(programação extrema)** → recente evolução da abordagem incremental

- Tem como base o desenvolvimento e a entrega de incrementos de funcionalidade muito pequenos, o envolvimento do usuário no processo, a constante melhoria de código e a programação impessoal

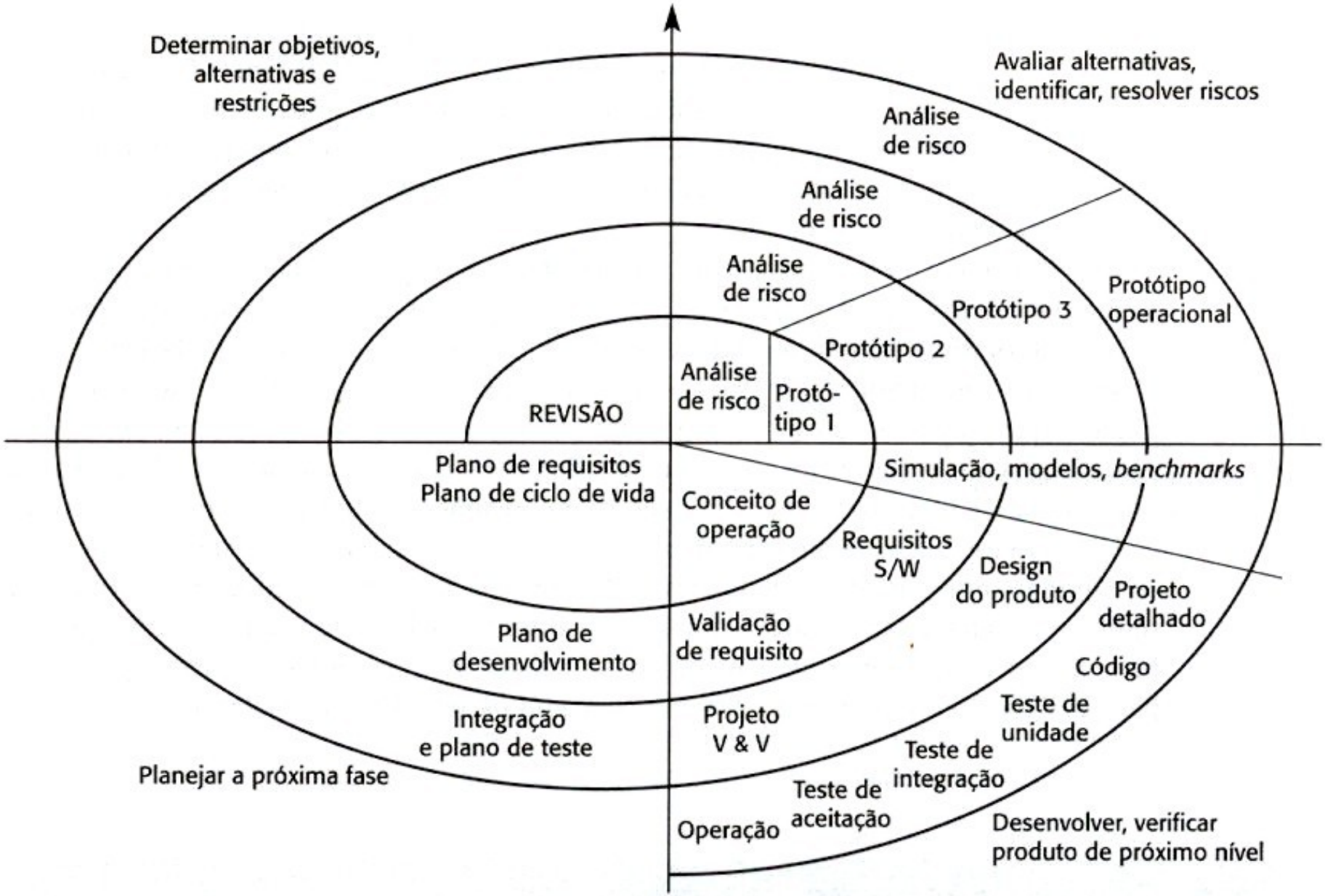


Modelo em Espiral

Em vez de representar o processo de *software* como uma sequência de atividades com algum retorno de uma atividade para outra, o processo é representado como uma espiral

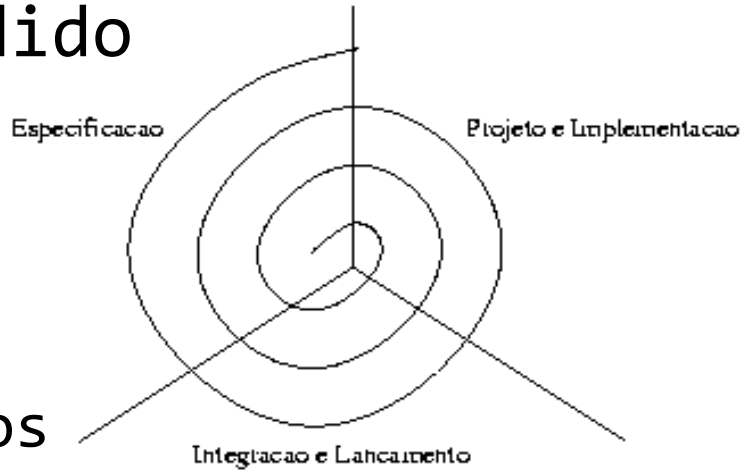
- Cada *loop* na espiral representa uma fase do processo de *software*
- Assim, o *loop* mais interno pode estar relacionado à viabilidade do *software*
- O *loop* seguinte, à definição de requisitos para o *software*
- O próximo *loop*, ao projeto do *software*, e assim por diante...

Modelo em Espiral



Modelo em Espiral

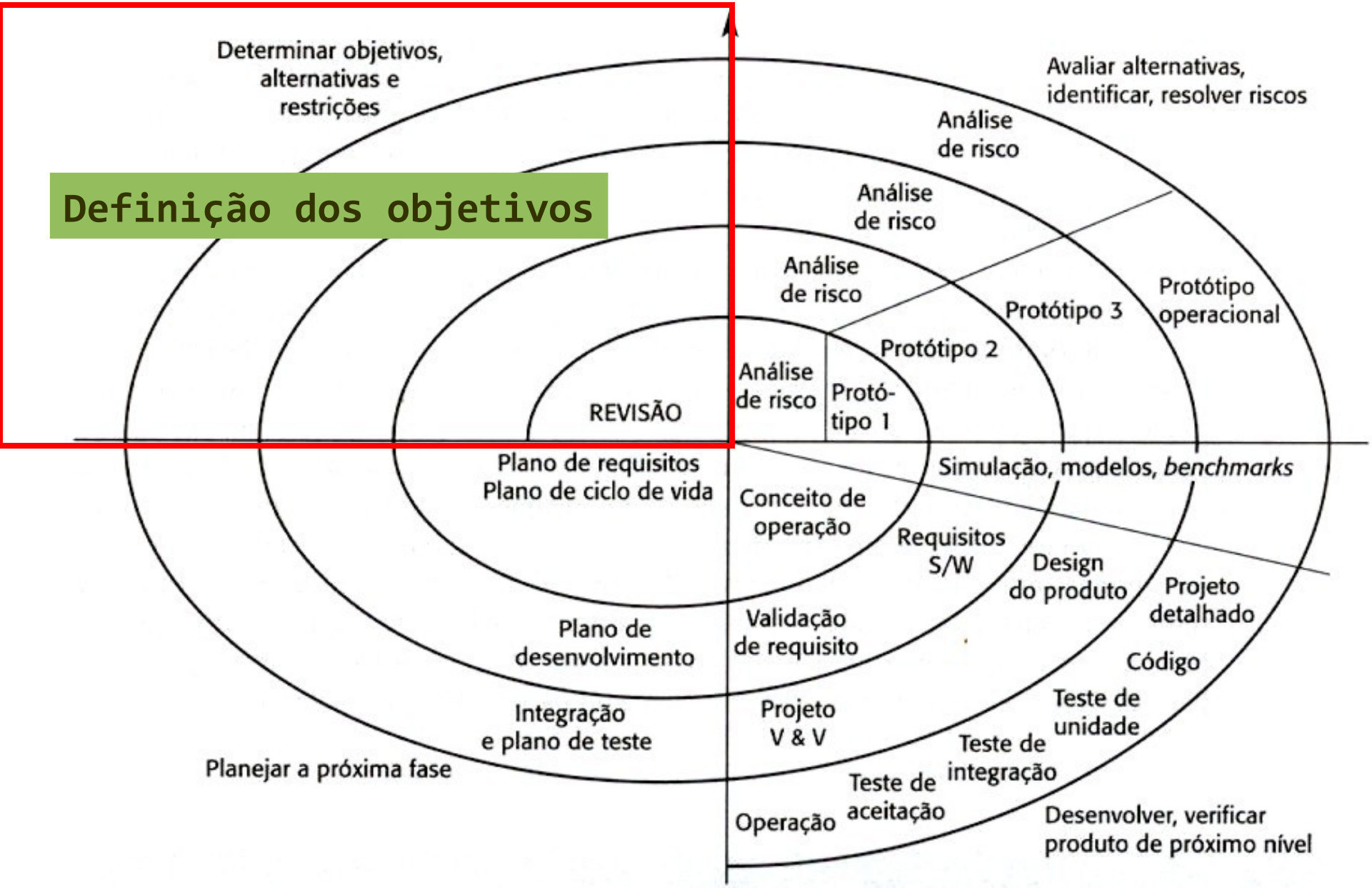
- Cada *loop* da espiral é dividido em 4 setores:



1. Definição de objetivos →

- São definidos os objetivos específicos para essa fase do projeto
- São identificados os riscos do projeto e, dependendo dos riscos poderão ser planejadas estratégias alternativas

Desenvolvimento em Espiral

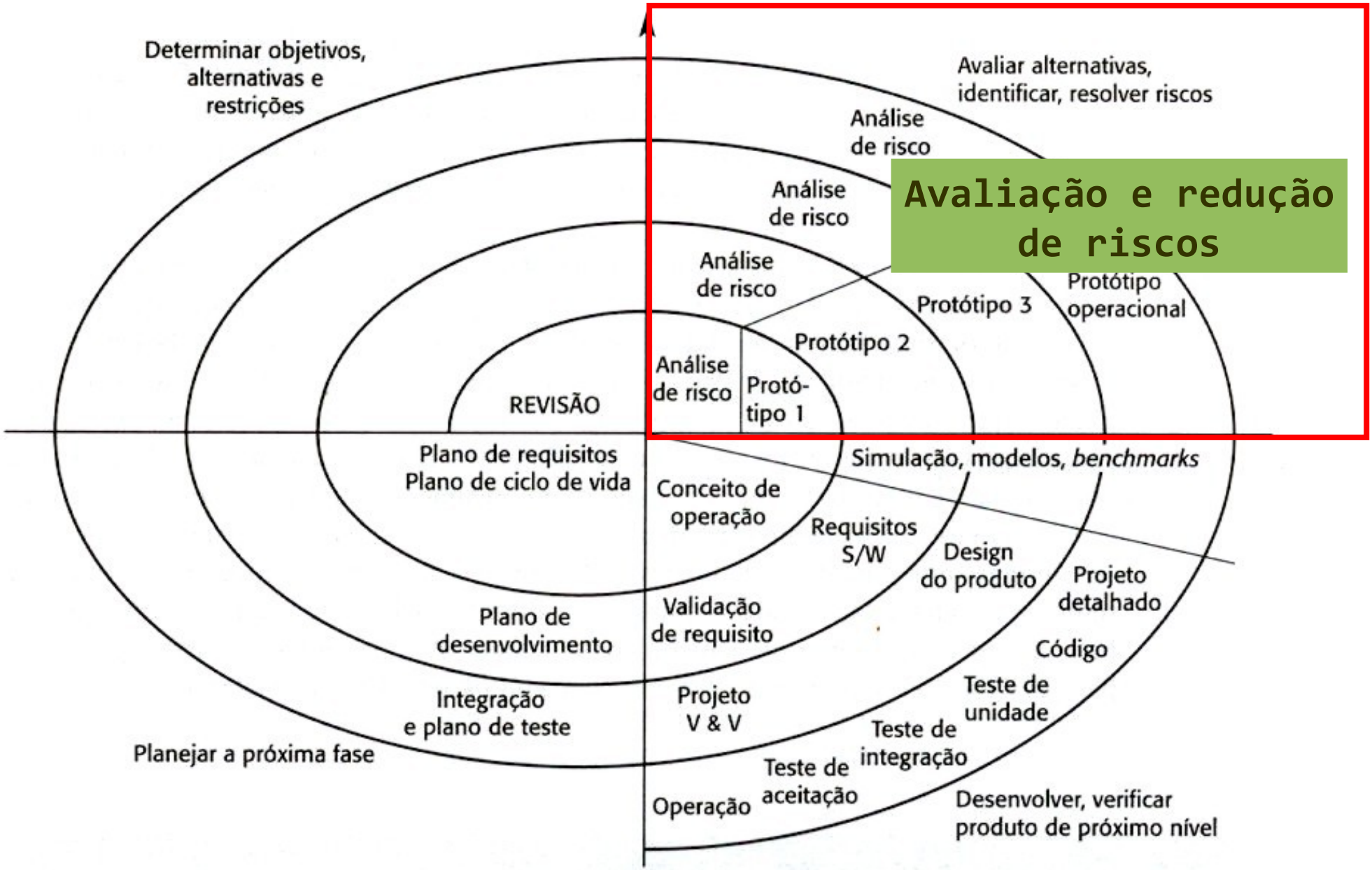


Desenvolvimento em Espiral

2. Avaliação e redução de riscos

- Para cada um dos riscos de projeto identificados, é realizada uma análise detalhada e são tomadas providências para reduzir esses riscos
- Por exemplo, se houver um risco de os requisitos serem inadequados, poderá ser desenvolvido um protótipo

Desenvolvimento em Espiral



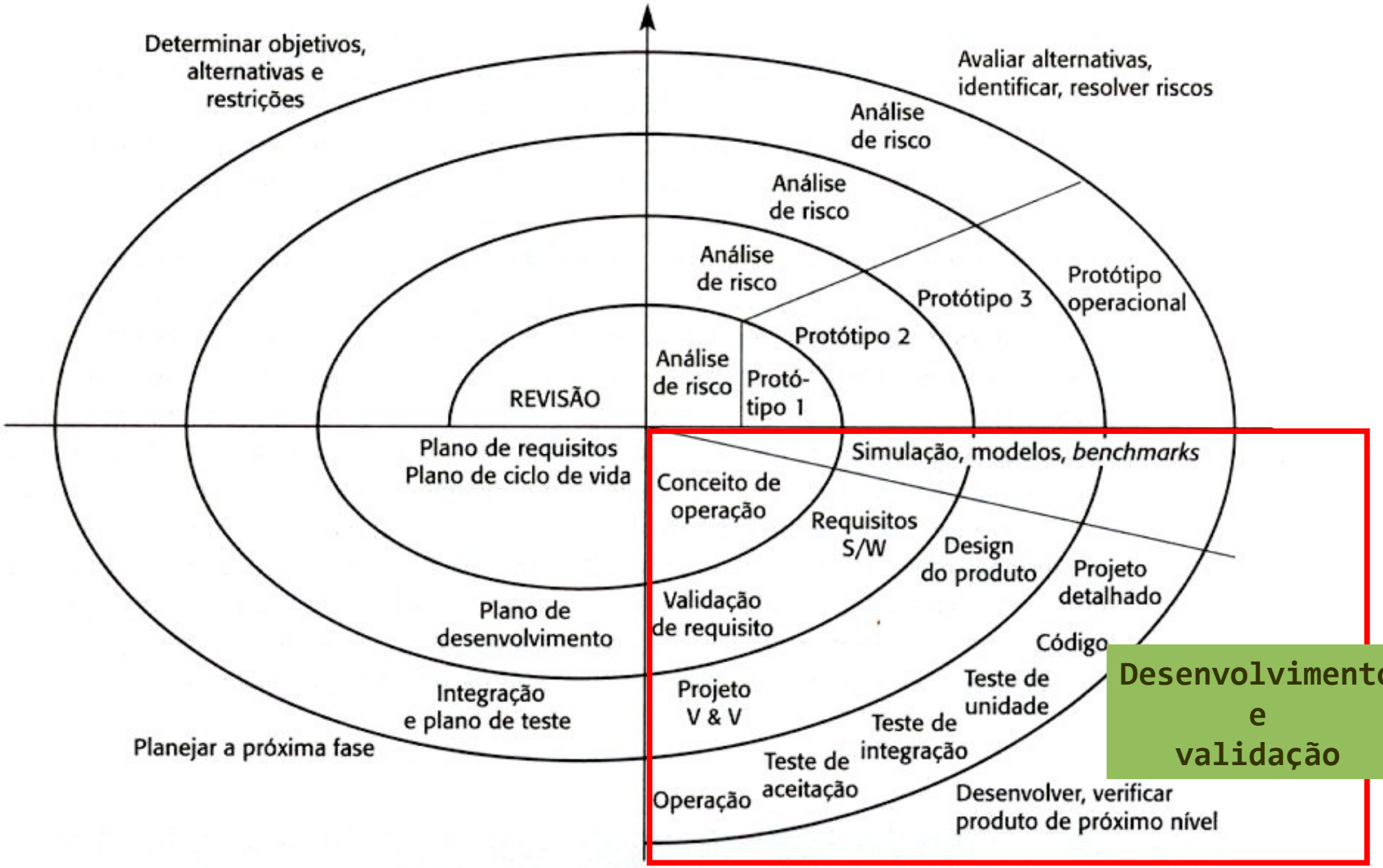
Avaliação e redução de riscos

Desenvolvimento em Espiral

3. Desenvolvimento e validação

- Depois da avaliação dos riscos, é escolhido um modelo de desenvolvimento para o *software*
- Por exemplo, se forem dominantes os riscos relacionados à interface com o usuário, pode ser utilizado o modelo de desenvolvimento evolucionário (prototipação)

Desenvolvimento em Espiral



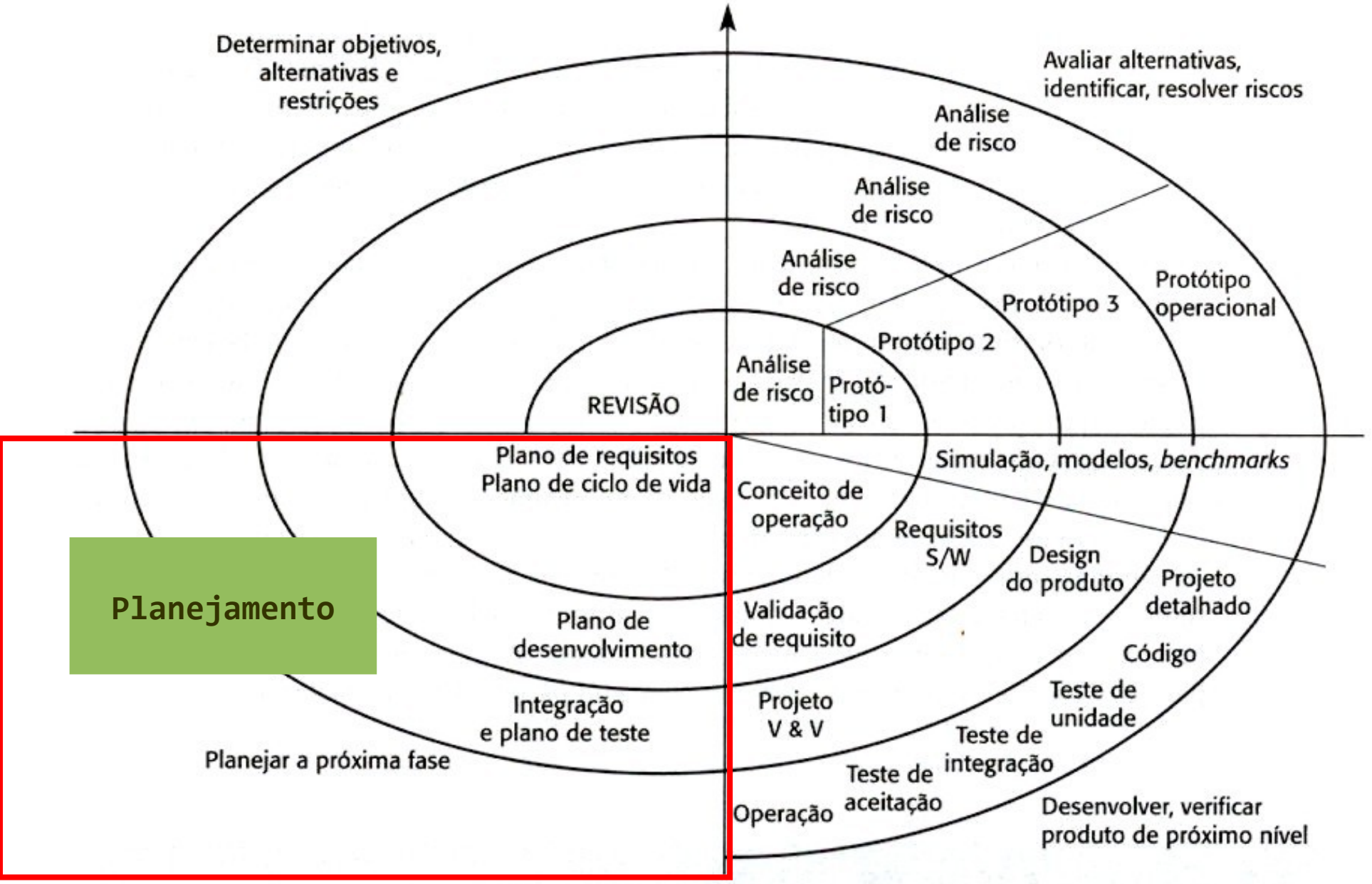
Desenvolvimento e validação

Desenvolvimento em Espiral

4. Planejamento

- O projeto é revisto e é tomada a decisão sobre continuar com o próximo *loop* da espiral
- Se a decisão for continuar, serão traçados os planos para a próxima fase do projeto

Desenvolvimento em Espiral



Desenvolvimento em Espiral

- Não há fases fixas, como especificação ou projeto, no modelo em espiral
- O modelo em espiral abrange outros modelos de processo, como por exemplo, prototipação
- Diferença do modelo em espiral em relação a outros modelos de processo de *software* → **explícita consideração dos riscos no modelo em espiral**

Etapas do processo de *software*

Especificação

Projeto

Implementação

Validação

Evolução

Especificação de *Software*

- Estabelece quais funções são requeridas pelo *software* e as restrições sobre sua operação e seu desenvolvimento

- Essa etapa é chamada também de Engenharia de Requisitos → de extrema importância

- O processo de engenharia de requisitos leva à produção do documento de requisitos, que é a especificação para o *software*

- Existem 4 atividades principais no processo de engenharia de requisitos → veremos a seguir...

Especificação de *Software*

1. Estudo de Viabilidade → existe tecnologia atual para o desenvolvimento do sistema? Existem restrições orçamentárias

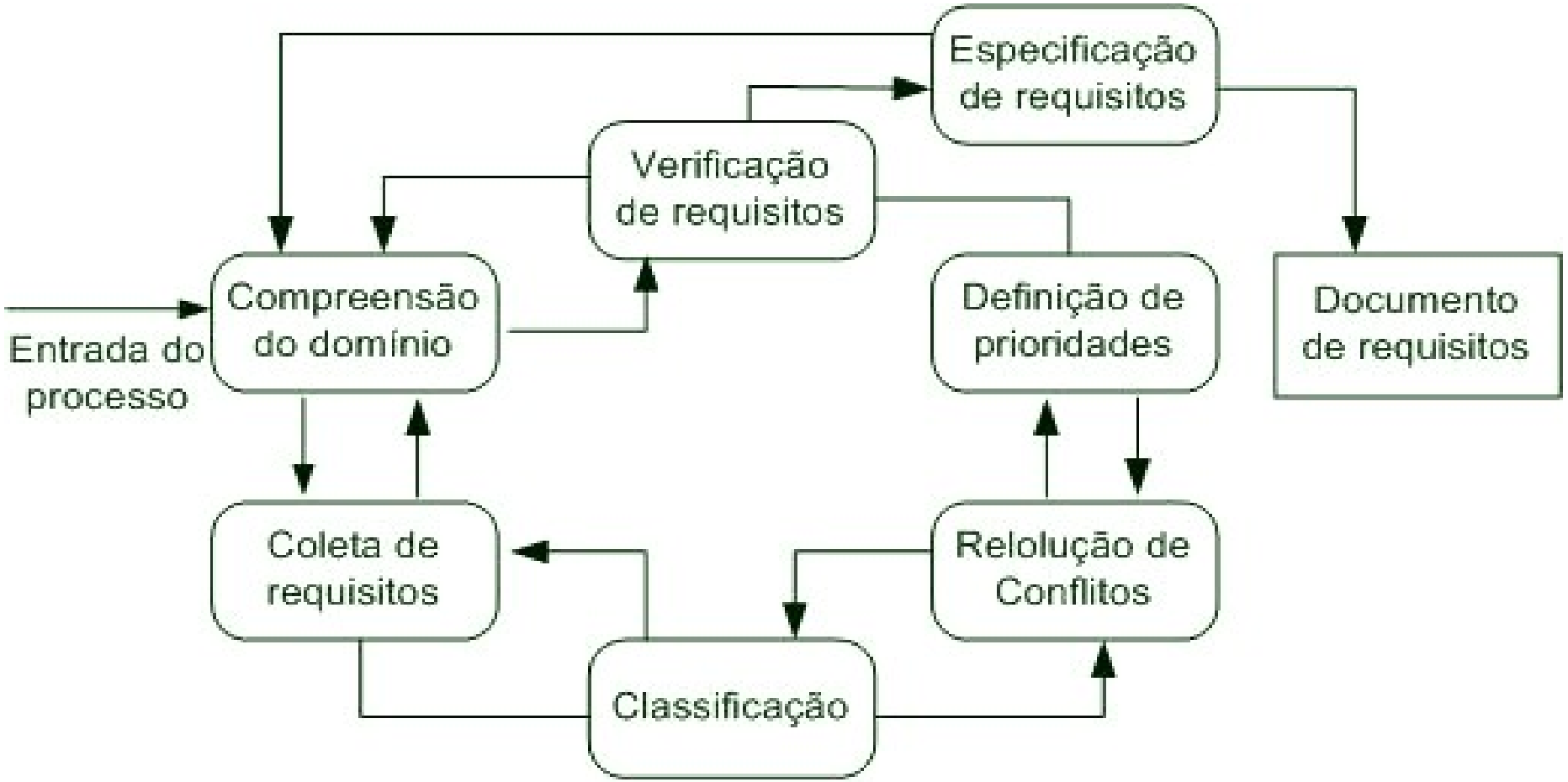
2. Levantamento e análise de requisitos → obtenção dos requisitos do sistema. Entrevista, observação, sistemas existentes, ...

3. Especificação de requisitos → documento que especifica os requisitos

4. Validação de requisitos → verificação dos requisitos quanto a pertinência, consistência e integralidade

Especificação de *Software*

Exemplo de fluxos de atividades da etapa de Especificação de *Software* ou Engenharia de Requisitos



Projeto e Implementação de *Software*

Um projeto de *software* é uma descrição estruturada a ser implementada, dos dados que são parte do sistema, das interfaces entre os elementos do sistema e dos algoritmos utilizado

- **Métodos de Projeto →**
  - Projeto estruturado
  - Projeto orientado a objetos

Projeto e Implementação de *Software*

- **Codificação**
  - Costuma ser individualizada, sem regras e dependente da criatividade e competência pessoal

- **Teste x Depuração**
  - Teste → estabelece a existência de defeitos
  - Depuração → localiza e corrige esses defeitos

Validação de *Software*

- Destina-se a mostrar que um *software* está de acordo com suas especificações e que atende às expectativas do usuário

- Processo de teste
  1. **Teste de unidade** → componentes individuais
  2. **Teste de módulo** → coleção de componentes
  3. **Teste de subsistema** → conjunto de módulos integrados
  4. **Teste de sistema** → integração dos subsistemas
  5. **Teste de aceitação** → o *software* é testado com os dados fornecidos pelo usuário, no lugar dos testes simulados

Evolução de Software

- **Manutenção de *software***

→ é o processo de modificar o *software* desenvolvido depois que o mesmo é colocado em operação

→ pode acontecer motivado por identificação de erros no *software* ou por novos requisitos

O *software* pode ser continuamente modificado ao longo de seu tempo de duração, em resposta a requisitos em constante modificação e às necessidades do usuário

Ferramentas CASE

- É o nome dado ao *software* utilizado para apoiar as atividades de processo de software, como a engenharia de requisitos, o projeto, o desenvolvimento e os testes

- As ferramentas CASE, portanto, incluem editores de projeto, dicionários de dados, compiladores, depuradores, ferramentas de construção de sistemas, entre outros

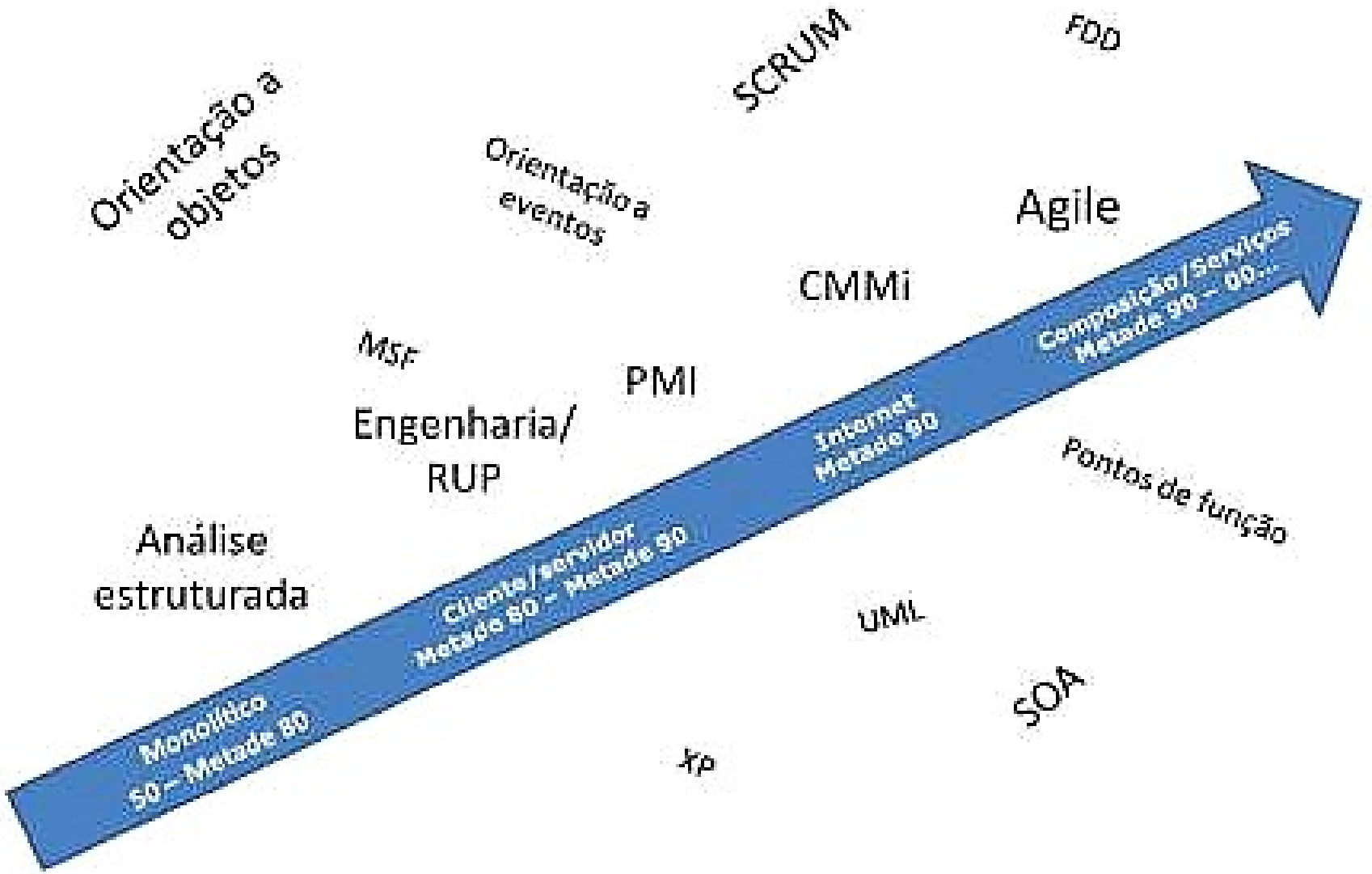
Ferramentas CASE

- **Exemplos de atividades que podem ser automatizadas utilizando-se CASE**
  - O desenvolvimento de modelos gráficos de *software*, como parte das especificações de requisitos ou do projeto de *software*
  - A compreensão de um projeto utilizando-se um dicionário de dados que contém informações sobre as entidades/objetos
  - A geração de interfaces com usuários

Ferramentas CASE

- **Exemplos de atividades que podem ser automatizadas utilizando-se ferramentas CASE**
  - A depuração de código, pelo fornecimento de informações sobre um programa em execução
  - Tradução automatizada de programas, a partir de uma antiga versão de uma linguagem de programação, como Cobol, para uma versão mais recente

Evolução dos processos de *software*



Para praticar...

Considerando os modelos de processos de *software* discutidos aqui, qual a melhor opção para qual tipo de *software*?

Absorvendo as melhores práticas de cada modelo, como poderia ser um modelo ótimo?

O que acham de elaborar um quadro comparativo?

Consultar: Pressman (2011), p. 64

Para refletir...

O texto apresentado por André Nascimento, em seu blog (http://blog.anascimento.net/tag/fabrica-de-software/) faz uma análise sobre as fábricas de *software*.

O que pensam sobre o ponto de vista do autor?

Referências

- PRESMANN, R. **Engenharia de Software**: uma abordagem profissional. 7. ed. Rio de Janeiro: Mc Graw Hill, 2011. Cap. 2
- SOMMERVILLE, I. **Engenharia de Software**. 8. ed. Rio de Janeiro: Pearson, 2007. Cap. 4

Sugestão de consulta/leitura:
http://julianakolb.com/category/engenharia-de-software/sumario-engenharia-de-software/

Vídeos sugeridos

- Aula 2 - Engenharia de *Software* (Processo de desenvolvimento de *software*)

  -> aulas desenvolvidas por alunos

  - Parte 01 - www.youtube.com/watch?v=igeU6B5GmIU
  - Parte 02 - www.youtube.com/watch?v=0fRv6o3aakw

- Ver também os vídeos indicados no material sobre Ciclo de Vida de Sistemas de Informação