

ENGENHARIA DE SOFTWARE



Aula 3

Professor.: Prof. Me. João Paulo Biazotto



Tópicos da aula

- Processos de Software
- Modelos de Processos de Software
- Atividades Básicas do Processo de Software



Introdução

Como posso iniciar o desenvolvimento de um software?

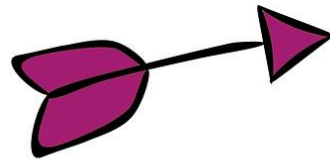


Processo

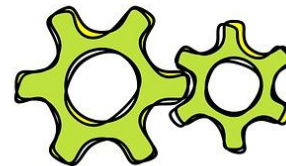
Um conjunto de atividades que recebe insumos e os transforma, seguindo uma lógica pré-estabelecida e agregando valor, em produtos ou serviços destinados a atender às necessidades dos clientes ou usuários.



Idea



To do



Doing



Done

Processo

- ❑ O Guia PMBOK® define processo como sendo um **conjunto de atividades inter-relacionadas realizadas para obter um conjunto específico de produtos, resultados ou serviços**[2] (PMBOK, 2008).
- ❑ Segundo o IEEE[3], um processo é uma **sequência de passos executada com um determinado objetivo** (IEEE, 2003).



Processo

Para o CMMI[4], um processo é definido quando tem uma descrição que é mantida, ou seja, tem documentação que detalha o que é feito (produto), quando (etapas), por quem (papéis), os itens utilizados (insumos) e os itens produzidos (resultados)(CMMI, 2006).

Os processos podem ser definidos com mais ou menos detalhes e suas etapas podem ter ordenação parcial, o que pode permitir paralelismo entre algumas delas (PAULA FILHO, 2009).



Processo de Software

Objetivos:

- Estabelecer as atividades a serem realizadas durante o projeto.
- Determinar quando, como e por quem essas atividades serão executadas.



Quais ou o que?



Quando?



Como?





Processo de Software

A aplicação de uma abordagem sistemática, disciplinada e possível de ser medida para o desenvolvimento, operação e manutenção do software (IEEE).

Processo de Software - Atividades

Conjunto de etapas composto por diversas tarefas com o objetivo de desenvolver um software específico.

As etapas para desenvolvimento de um software são:

- Especificação;
- Projeto e implementação;
- Validação;
- Evolução;



Processo de Software

- **Especificação:** Defina a funcionalidade do software e as restrições sobre sua operação.
- **Projeto e implementação:** O software deve ser desenvolvido conforme a especificação.
- **Validação:** O software deve ser validado para assegurar que atenda às expectativas do cliente.
- **Evolução:** O software deve evoluir para atender aos novos requisitos que surgirem.



Processo de Software

- **Especificação:** Defina a funcionalidade do software e as restrições sobre sua operação. (Requisitos)
- **Projeto e implementação:** O software deve ser desenvolvido conforme a especificação. (Modelos e codificação)
- **Validação:** O software deve ser validado para assegurar que atenda às expectativas do cliente. (Testes)
- **Evolução:** O software deve evoluir para atender aos novos requisitos que surgirem. (Modificações)



Especificação

- **Estudo de Viabilidade:** Avalia as condições comerciais e orçamentárias.
- **Levantamento e Análise de Requisitos:** Envolve conversas, entrevistas e prototipação.
- **Especificação de Requisitos:** Tradução e filtragem dos requisitos identificados.
- **Validação de requisitos:** Verifica a pertinência, consistência e integralidade dos requisitos.



Projeto e Implementação

- O projeto de software consiste em criar um modelo detalhado do sistema, abrangendo sua arquitetura, estruturas de dados, interfaces e componentes.
- Na implementação, o sistema é traduzido da descrição computacional da fase de projeto para código executável, utilizando uma ou mais linguagens de programação.



Validação

- Visa demonstrar que um software cumpre suas especificações e atende às expectativas do usuário.
- Consiste em verificar cada etapa do processo, desde a definição dos requisitos dos usuários até o desenvolvimento de cada programa que compõe o sistema.



Evolução

- O software evolui para atender às mudanças nas necessidades dos usuários;
- Após a implantação de um sistema, é inevitável que ocorram mudanças, como:
 - Pequenos ajustes pós-implantação
 - Melhorias substanciais
 - Adequação à legislação
 - Atendendo novos requisitos dos usuários
 - Correção de erros



Processo de Software - Artefatos

- Especificação de Software: Documento de Requisitos;
- Projeto e Implementação: Abrange desde a concepção do projeto até a implementação, englobando documentação, código-fonte e manuais do software;
- Validação de Software: Envolve testes, detecção e documentação de defeitos, garantindo a conformidade e qualidade do produto;
- Evolução de Software: Implementação de mudanças ou atualizações nos requisitos de software durante sua evolução;



Resumo

Durante o desenvolvimento de software:

- As **atividades principais** incluem especificação, projeto, implementação, validação e evolução.
- Os **artefatos resultantes** dessas atividades são documentos de requisitos e projeto, planos de testes, manuais e requisitos de evolução.





Ciclo de Vida de Software

Descreve a vida de um produto de software desde sua concepção até sua implementação, implantação, uso e manutenção.

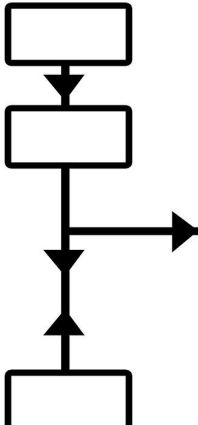


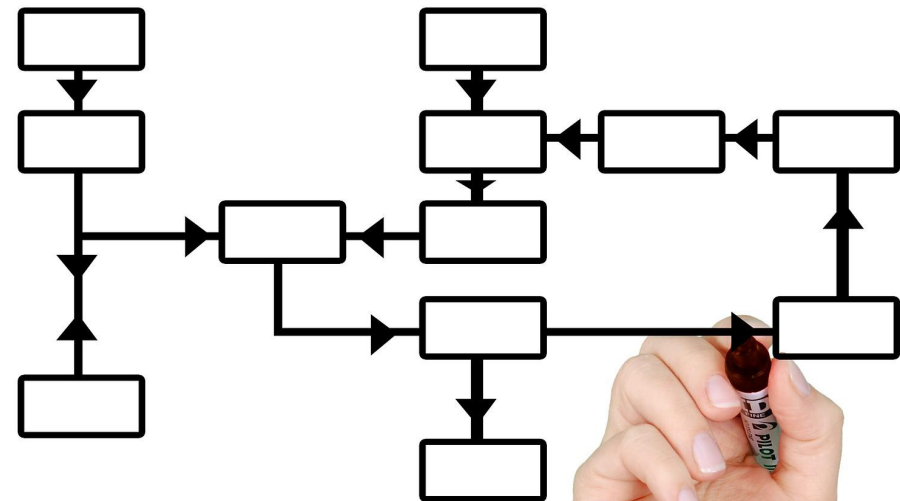
Ciclo de Vida de Software

Descrições abstratas dos estágios do desenvolvimento e manutenção de um software executável, geralmente ilustrando o processo de desenvolvimento e modificação.



Modelos de Ciclo de Vida de Software

- O Modelo Cascata (ciclo de vida clássico);
 - O Modelo de Prototipação;
 - O Modelo Incremental;
 - O Modelo Espiral;
 - Outros modelos;
- 
- ```
graph TD; A[] --> B[]; B --> C[]; C --> B;
```
- O diagrama ilustra o Modelo Cascata, também conhecido como ciclo de vida clássico. Ele é representado por uma sequência vertical de três retângulos. O primeiro retângulo no topo aponta para o segundo retângulo no meio por uma seta descendente. O segundo retângulo aponta para o terceiro retângulo na base por uma seta descendente. Além disso, há uma seta horizontal que sai do lado direito do segundo retângulo e aponta para a direita, representando uma saída ou entrega. Uma seta ascendente também aponta do terceiro retângulo de volta para o segundo, indicando um ciclo de feedback ou iteração.



# Modelos de Ciclo de vida de Software

## O Modelo Cascata



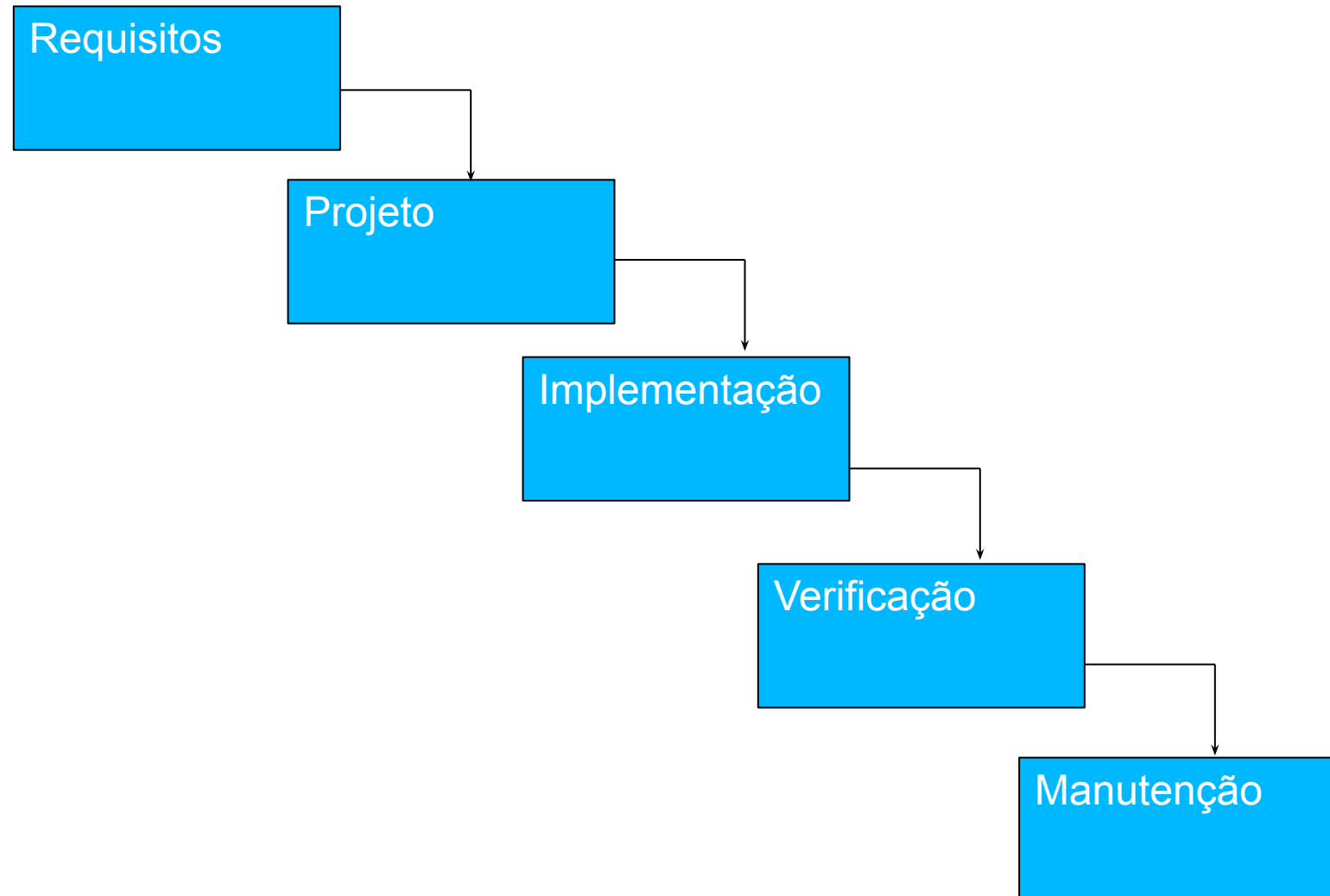
## Modelo Cascata

O Modelo Cascata é empregado principalmente quando há um claro entendimento dos requisitos do problema em questão.

Exemplo:

Quando há necessidade de realizar ajustes ou melhorias em um sistema já estabelecido, como adaptações devido a mudanças ou criações de leis governamentais.

# Modelo Cascata





## Modelo Cascata

Modelo propõe uma **abordagem metodológica sequencial** e estruturada para o desenvolvimento de software:

**Requisitos** - Iniciamos com a elicitação de requisitos;

**Projeto** – Planejamento, incluindo estimativas, cronograma e acompanhamento;

**Implementação** – Modelagem com análise e design;

**Verificação** - Construção da codificação e testes;

**Manutenção** - Implantação ou implementação aqui temos entrega, suporte e obtenção de feedback sobre o software finalizado.

## Modelo Cascata

O modelo cascata, o paradigma **mais antigo da engenharia de software**, continua sendo amplamente utilizado na indústria, apesar de receber críticas significativas que questionam sua eficácia, inclusive por parte de seus proponentes mais fervorosos.



## Modelo Cascata

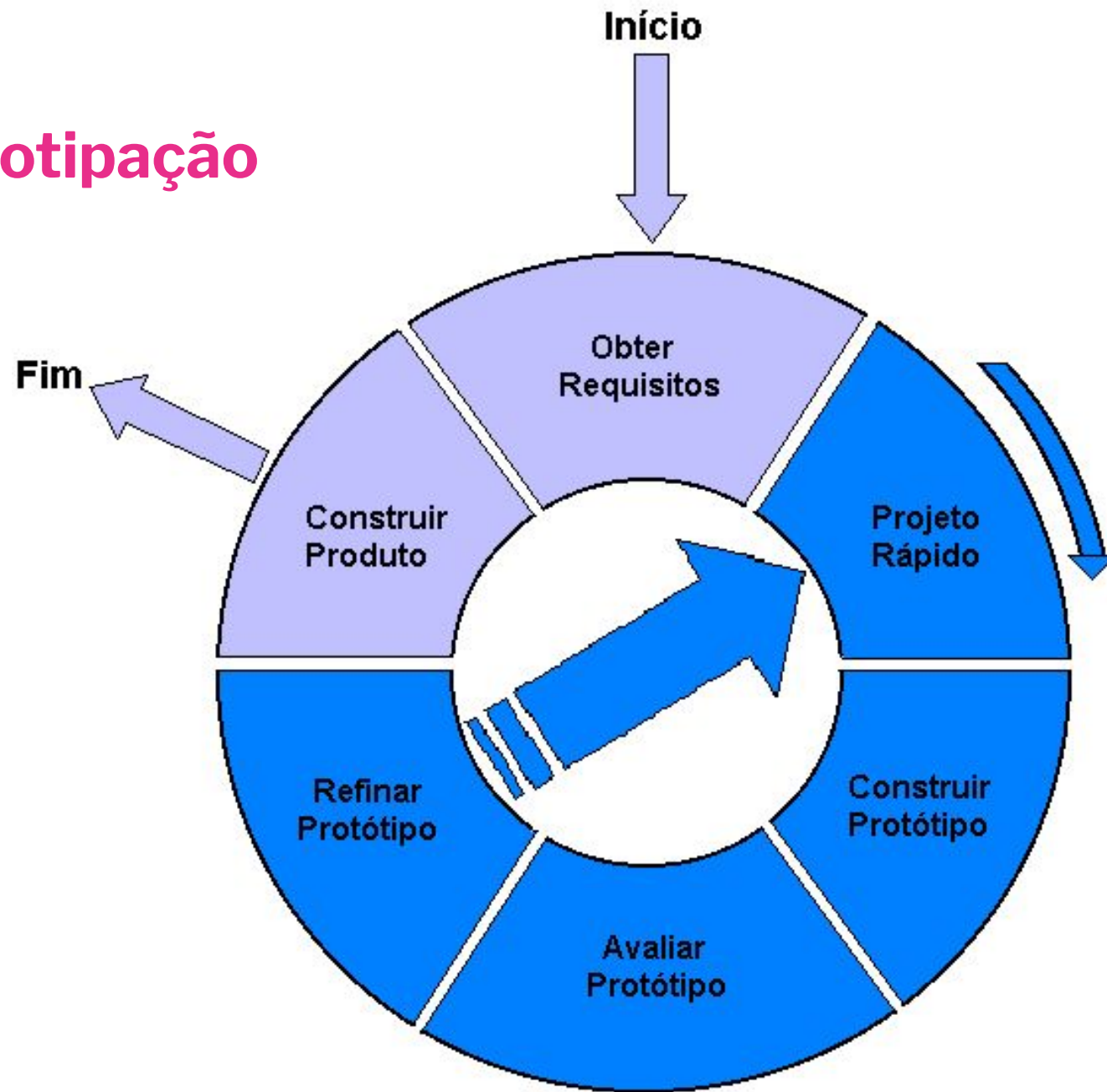
Principais “desafios” no modelo cascata incluem:

1. Projetos de software na indústria raramente seguem um fluxo sequencial.
2. É incomum e desafiador para o cliente explicitar todas as suas necessidades desde o início.
3. Uma versão operacional pronta para o cliente só estará disponível próximo ao término do projeto.
4. Um erro grave nas etapas iniciais pode resultar em um projeto desastroso.
5. Equipe aguardando.

## Modelo de Prototipação

- O objetivo é compreender as necessidades do usuário para definir de forma mais precisa os requisitos do sistema;
- Permite ao desenvolvedor criar um modelo de software que corresponda às expectativas
- Especialmente útil quando o cliente não especificou detalhadamente os requisitos.

# Modelo de Prototipação



# Modelo de Prototipação

- **Obter Requisitos:**

O desenvolvedor e o cliente estabelecem os objetivos principais do software, identificam os requisitos já conhecidos e as áreas que requerem definições adicionais.

- **Projeto Rápido | Construir Protótipo | Avaliar Protótipo:**

Exibição dos elementos de software perceptíveis ao usuário (interfaces de entrada e formatos de saída).



# Modelo de Prototipação

- **Refinar Protótipo**

Cliente e desenvolvedor refinam os requisitos do software a ser desenvolvido.

Neste ponto ocorre um processo de iteração que pode conduzir a atividade **Obter Requisitos** até que as necessidades do cliente sejam satisfeitas e o desenvolvedor compreenda o que precisa ser feito.

# Modelo de Prototipação

## Características

- Capacita os desenvolvedores a entender claramente o que deve ser criado;
- Aproxima o cliente do desenvolvimento;
- É ideal para quando o cliente não especificou detalhadamente os requisitos de entrada, processamento e saída.

# Modelo de Prototipação

- **Problemas**

- ☐ Frequentemente, ocorre confusão por parte do cliente entre o protótipo e o produto final, o que pode ser prejudicial, já que o protótipo não alcança a qualidade desejada.
- ☐ O desenvolvedor implementa utilizando apenas recursos disponíveis para prototipar rapidamente.

# Modelo de Prototipação

- Problemas

- ☐ O cliente não está ciente de que o software que ele vê não foi desenvolvido levando em consideração a qualidade geral e a facilidade de manutenção a longo prazo.
- ☐ Não aceita a ideia que a versão final do software será desenvolvida e insiste em utilizar o do protótipo como produto final;

## Os Modelos Evolutivos

- ❑ Quando os requisitos do produto e de negócio mudam ao longo do desenvolvimento;
- ❑ Quando o prazo de entrega é apertado (Demanda do mercado), torna-se inviável concluir o produto por completo;
- ❑ Quando os requisitos principais são bem conhecidos, mas os detalhes ainda precisam ser definidos;

## Os Modelos Evolutivos

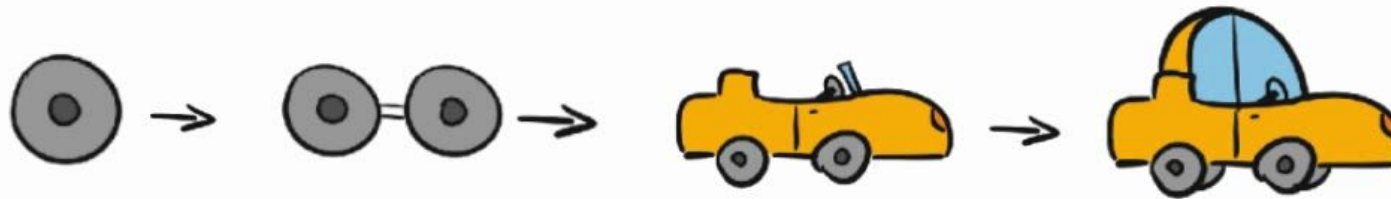
- ❑ Modelos evolutivos são iterativos;
- ❑ Possibilitam o desenvolvimento de versões cada vez mais completas do software;
- ❑ A cada iteração, se avança no conhecimento do projeto, novos requisitos são elicitados e a arquitetura do software é revisada.



## Modelo Incremental

- O modelo incremental mescla elementos do modelo cascata, aplicado repetidamente, com a filosofia iterativa da prototipação.
- O objetivo é colaborar com o usuário para identificar seus requisitos de forma incremental, até alcançar o produto final.
- Um processo de desenvolvimento de software é considerado incremental quando, a cada etapa, uma parte completa do software é desenvolvida.

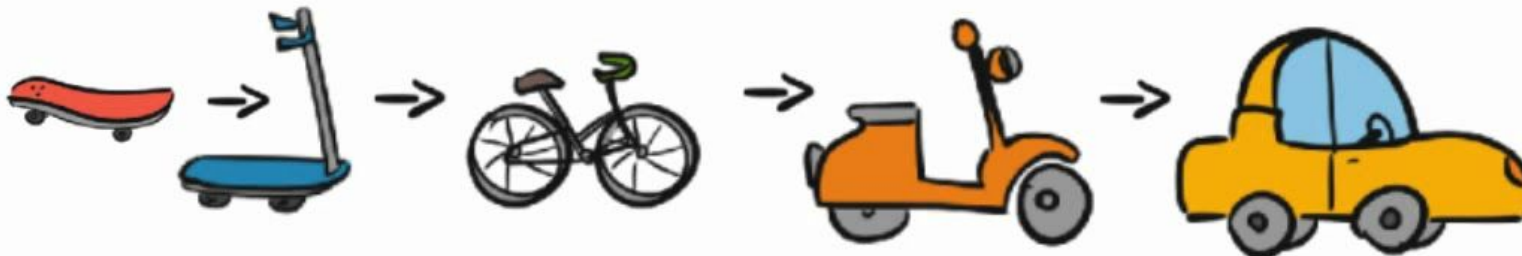
# Modelo Incremental



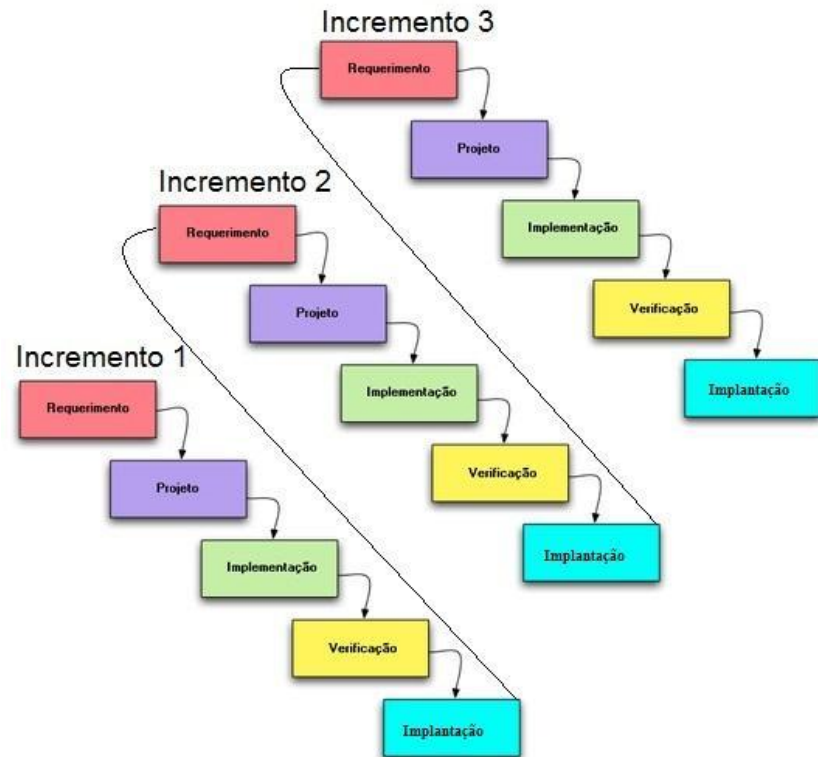
MÓDELO ITERATIVO

---

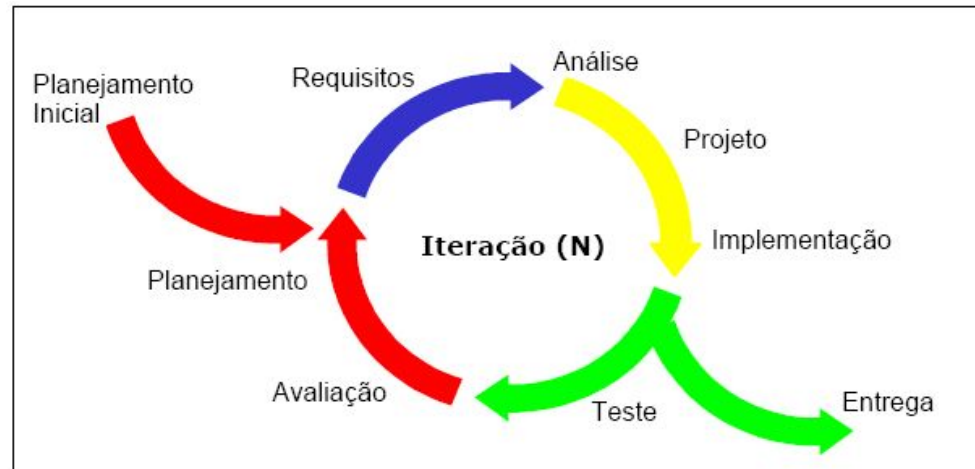
MÓDELO ITERATIVO E INCREMENTAL



# Modelo Incremental



# Modelo Iterativo Incremental



1



2



3



## Modelo Incremental

- ❑ A versão inicial geralmente é o núcleo do produto, sendo sua parte mais importante;
- ❑ A evolução ocorre com a adição de novas características sugeridas pelo usuário;
- ❑ Este modelo é essencial quando não é possível definir detalhadamente os requisitos antecipadamente;

## Modelo Incremental - Vantagens

- ✓ Permite mudanças nos requisitos baseadas no feedback após cada incremento;
- ✓ Identifica e resolve problemas cedo, minimizando grandes defeitos;
- ✓ Funcionalidades principais são entregues cedo, permitindo uso parcial do sistema antes da conclusão;
- ✓ Progresso medido com precisão após cada incremento concluído;



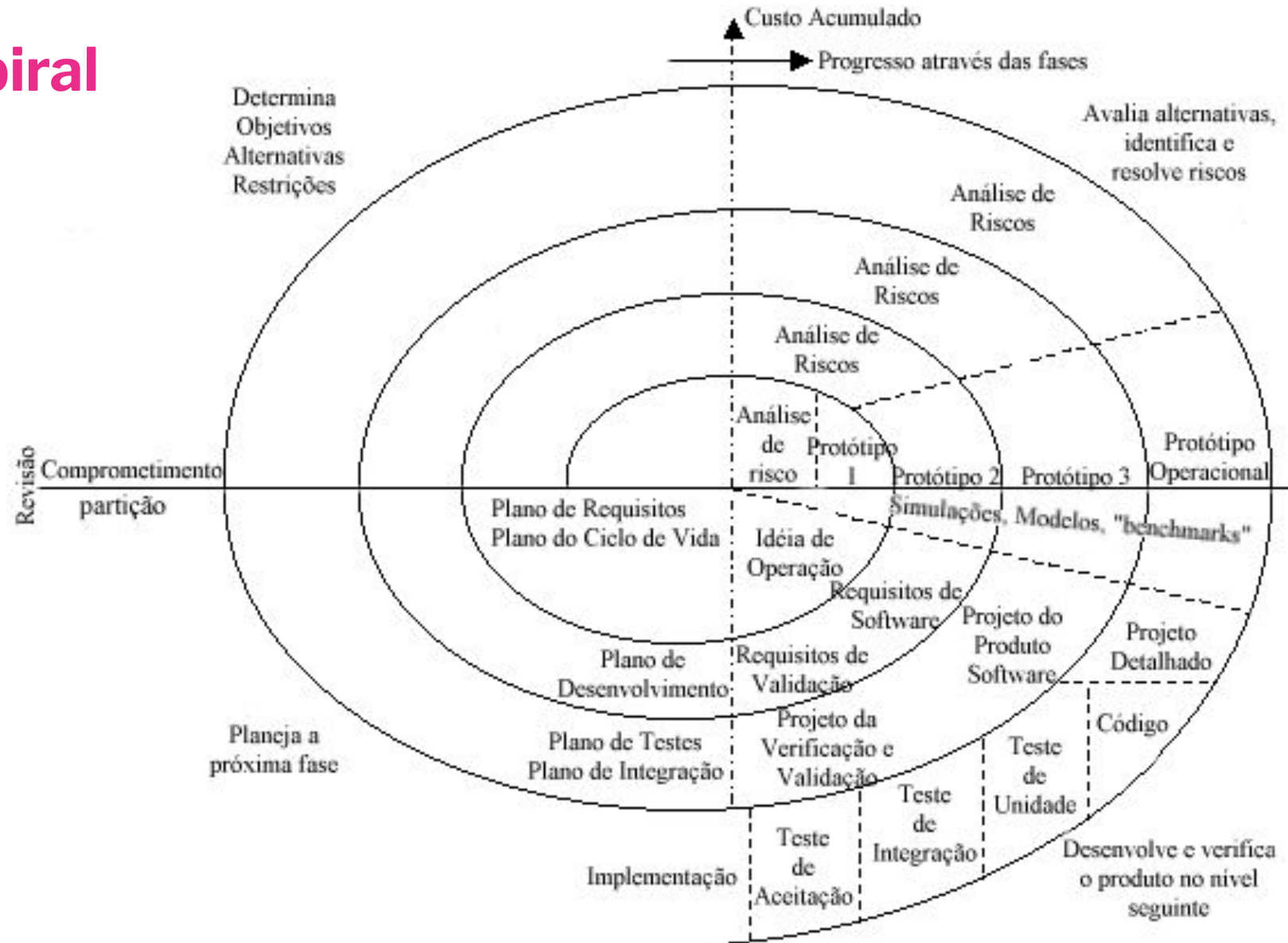
## Modelo Incremental – Desvantagens

- ❖ Incrementos podem ser mais complexos com mudanças frequentes nos requisitos;
- ❖ Desafiador sem bom gerenciamento contínuo de incrementos;
- ❖ Revisões e feedback contínuos podem sobrecarregar o projeto;

## Modelo Espiral

- ❑ O modelo espiral combina a natureza iterativa da prototipação com a abordagem controlada e sistemática do modelo cascata;
- ❑ O modelo espiral é subdividido em várias atividades ou áreas de tarefa;
- ❑ Existem tipicamente de 3 a 6 regiões de tarefa;

# Modelo Espiral



## Modelo Espiral

- Combina as melhores características do ciclo de vida clássico e da prototipação, incorporando também a análise de risco;
- Segue a abordagem de passos sistemáticos do ciclo de vida clássico;
- O produto é desenvolvido em uma série de iterações (voltas na espiral).

## Modelo Espiral

- ❑ Permite desenvolver produtos rapidamente, adicionando novas características e recursos de forma iterativa;
- ❑ Proporciona visibilidade ao cliente;
- ❑ Adota uma abordagem que permite ao desenvolvedor e ao cliente compreender e responder aos riscos em cada fase do processo;

## Modelo Espiral

- ❑ Pode ser desafiador convencer os clientes de que uma abordagem "evolutiva" é gerenciável;
- ❑ Requer experiência significativa na identificação de riscos, essencial para alcançar o sucesso;



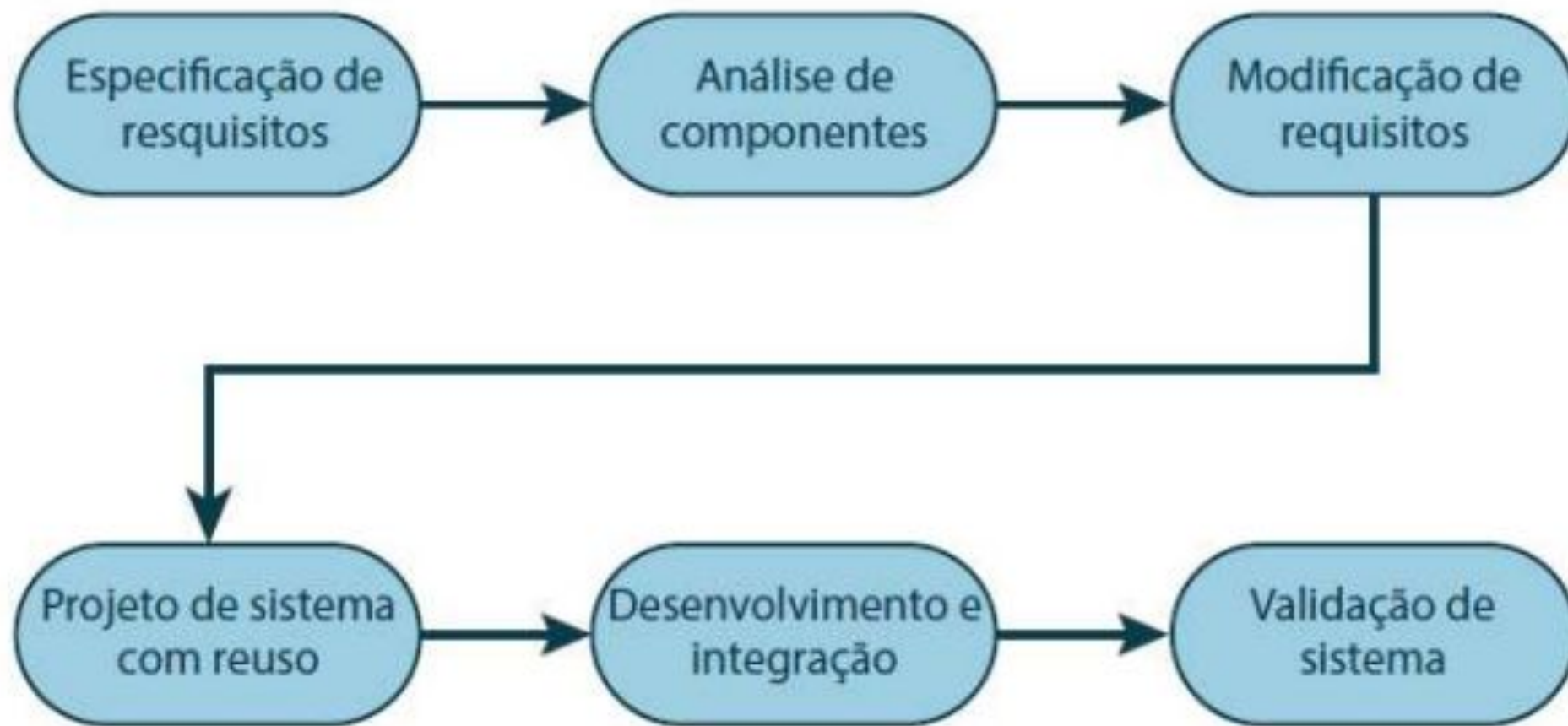
## Modelo Espiral x Modelo Incremental

- ❑ O Modelo Espiral é adequado para gerenciar projetos de grande escala;
- ❑ No modelo espiral, as fases não se sobrepõem;
- ❑ No modelo em espiral, é preciso contar com uma equipe ampla.

## Modelo de Reuso

- ✓ Aumento da Produtividade;
- ✓ Diminuição do tempo de desenvolvimento e validação
- Redução de custo;
- ✓ Qualidade dos Produtos;
- ✓ Flexibilidade na estrutura do software;
- ✓ Manutenibilidade;
- ✓ Familiaridade com o uso de padrões leva a menos erros.

## Modelo de Reuso



Fonte: Sommerville, 2011 (Adaptado)

## Modelo de Reuso

- **Análise de componentes:** Busca por componentes;
- **Modificação de requisitos:** Análise dos componentes encontrados;
- **Projeto do sistema com reuso:** Projeção do framework do sistema considerando seus componentes;
- **Desenvolvimento e Integração:** Integração entre o sistema e os sistemas adquiridos e analisados;