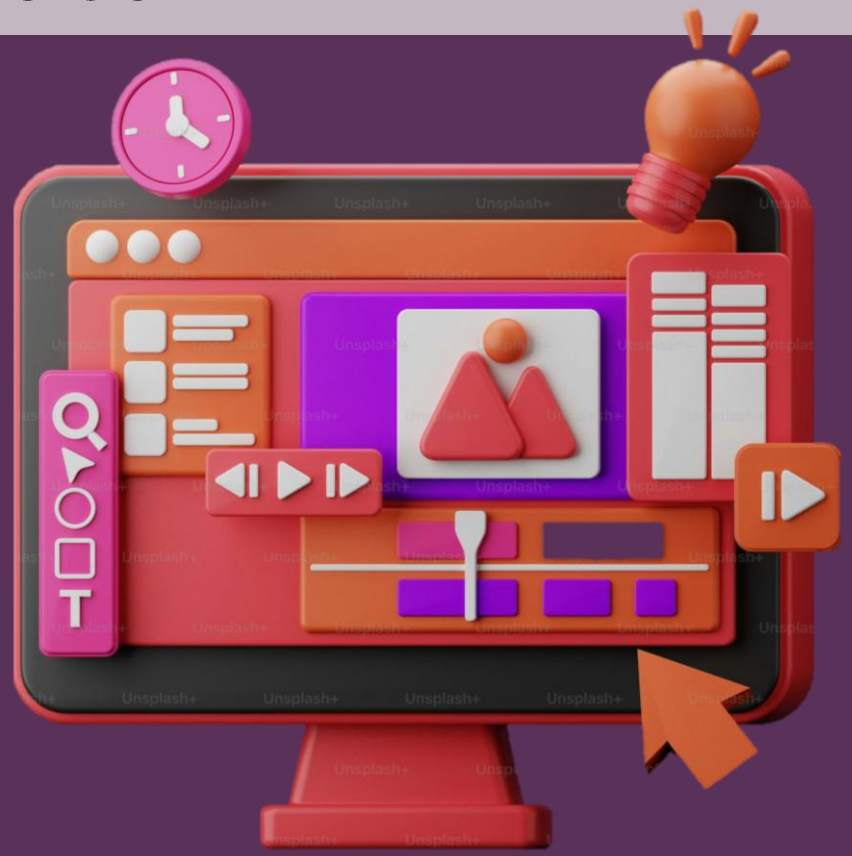
ENGENHARIA DE SOFTWARE

AULA 2 – MODELOS DE CICLO DE VIDA DO SOFTWARE



WALTER TRAVASSOS SARINHO

@PROF.WALTERTRAVASSOS

WALTER.TRAVASSOS@SECTRAS.EDU.BR

Lista de Exercícios 1

Aula de Hoje

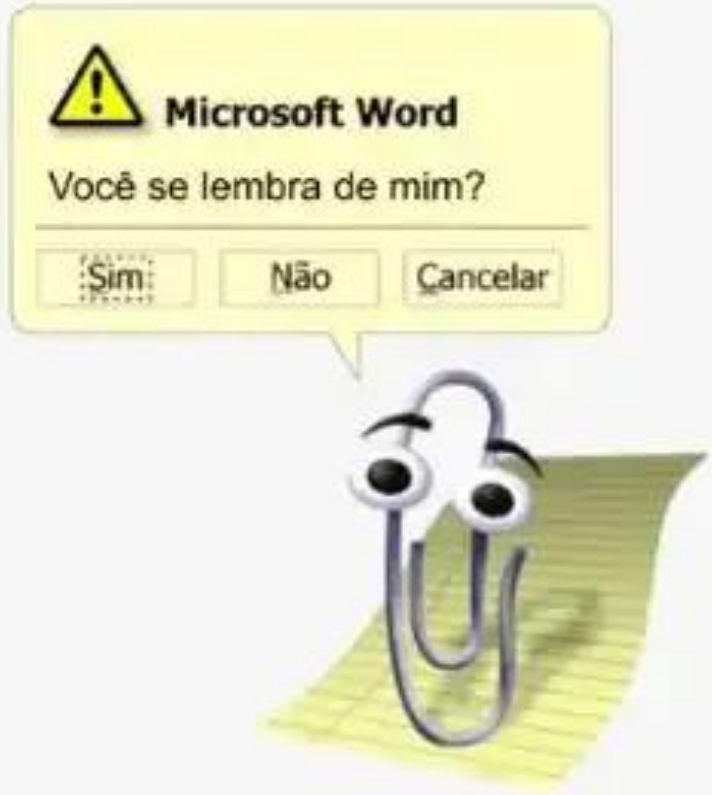
- O **ciclo de vida** do software.
- Apresentação de modelos de ciclo de vida do software.

Ciclo de Vida

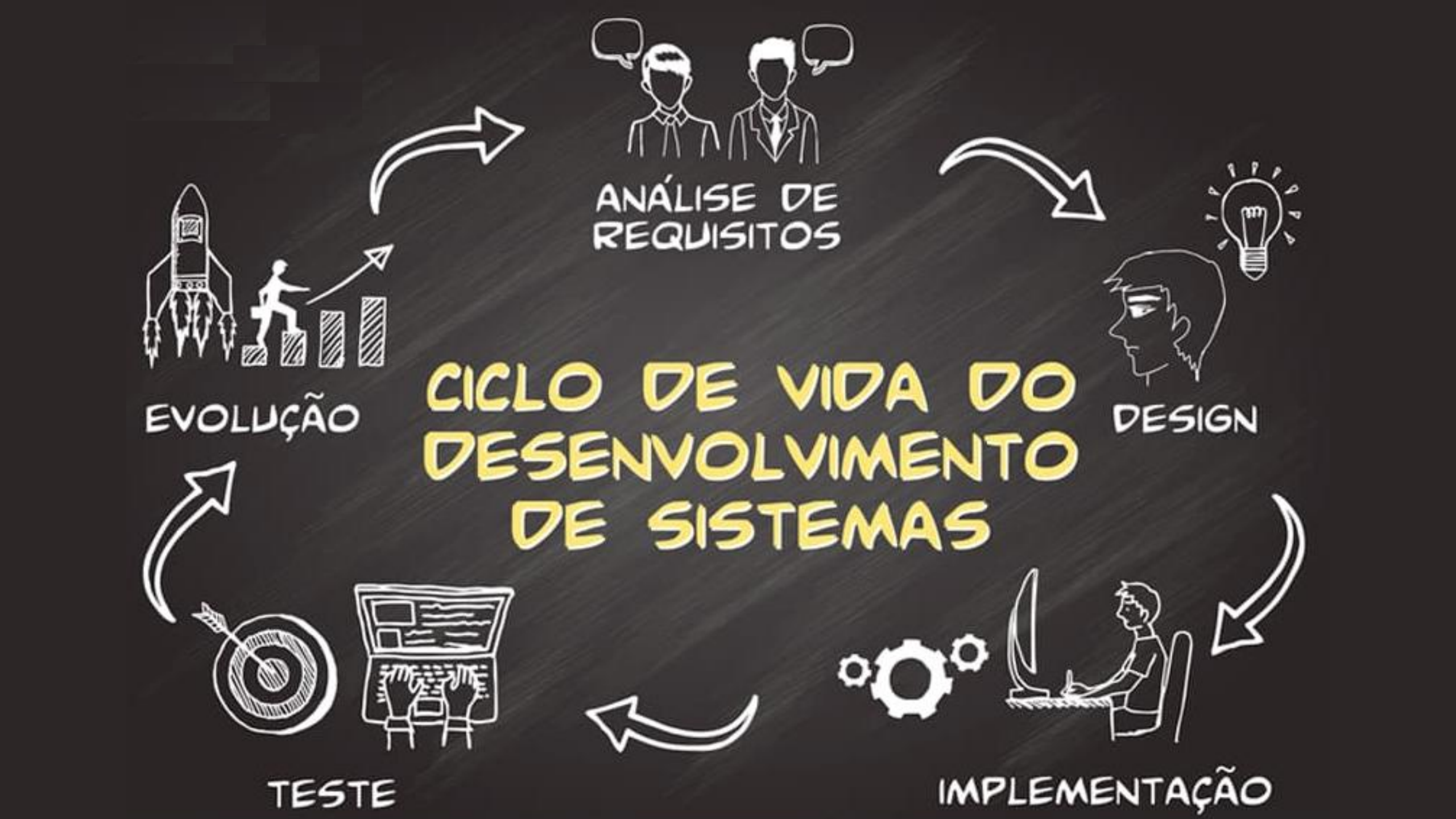
- Um **ciclo de vida** é um conceito que descreve a sequência de etapas ou fases que algo passa desde o seu início até o seu fim.
- O termo "ciclo de vida" pode ser aplicado a diversos contextos, como: produtos, sistemas, organismos vivos, projetos, e também, no software.

Ciclo de Vida do Software

- O ciclo de vida do software refere-se a todas as fases que um software passa desde sua concepção inicial até sua descontinuação.
- Esse ciclo cobre o processo completo de desenvolvimento e manutenção de software, abrangendo **desde a ideia inicial** até o **fim de sua vida útil**, incluindo as atualizações e a manutenção necessárias ao longo do tempo.



Principais Fases do Ciclo de Vida do Software


ANÁLISE DE REQUISITOS
DESIGN
IMPLEMENTAÇÃO
TESTE
EVOLUÇÃO
CICLO DE VIDA DO DESENVOLVIMENTO DE SISTEMAS

1. Planejamento e Análise de Requisitos

- **Descrição:** Esta fase envolve a coleta, análise e documentação dos requisitos do software. Os stakeholders (clientes, usuários finais, etc.) definem o que esperam do software, e esses requisitos são usados para guiar o desenvolvimento.
- **Atividades:** Levantamento de requisitos, análise de viabilidade, definição de escopo, documentação de requisitos.

2. Projeto de Software (Design)

- **Descrição:** Nesta fase, é criada a arquitetura do sistema e o design dos componentes do software, detalhando como o software será estruturado e como seus componentes irão interagir.
- **Atividades:** Projeto arquitetônico, design de interfaces, modelagem de dados, elaboração de diagramas (como diagramas de classes, de sequência, etc.).

3. Desenvolvimento (Implementação)

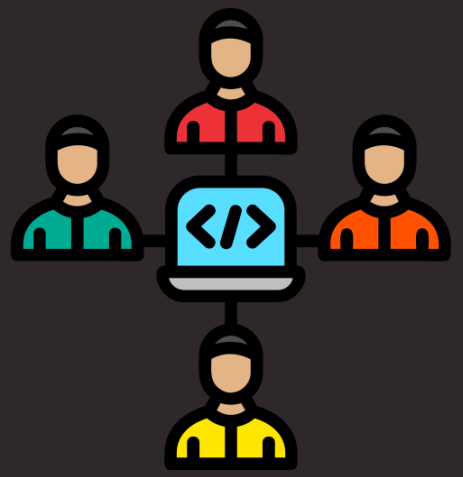
- **Descrição:** Consiste na codificação do software conforme os requisitos e o design estabelecidos nas fases anteriores. Os desenvolvedores escrevem o código-fonte e criam os componentes funcionais do software.
- **Atividades:** Programação, criação de testes unitários, integração de módulos.

4. Testes (Validação e Verificação)

- **Descrição:** O software é rigorosamente testado para garantir que atenda aos requisitos e funcione corretamente em diferentes condições. Isso inclui a verificação de bugs, a validação de funcionalidades e a garantia de que o software cumpre as expectativas dos usuários.
- **Atividades:** Testes unitários, testes de integração, testes de sistema, testes de aceitação, correção de bugs.

5. Implantação (Deployment)

- **Descrição:** Após os testes e a validação, o software é lançado para os usuários finais. Isso pode envolver a instalação do software em ambientes de produção, treinamento de usuários, e configuração de infraestrutura.
- **Atividades:** Instalação, configuração, treinamento de usuários, monitoramento inicial.



6. Manutenção e Suporte

- **Descrição:** Após a implantação, o software entra em uma fase de manutenção, onde é corrigido, atualizado e melhorado ao longo do tempo. Isso inclui a correção de defeitos, a adição de novas funcionalidades e a adaptação a mudanças tecnológicas ou de requisitos.
- **Atividades:** Correção de bugs, atualizações de software, melhorias de desempenho, suporte ao usuário.

7. Retirada de Serviço (Descontinuação)

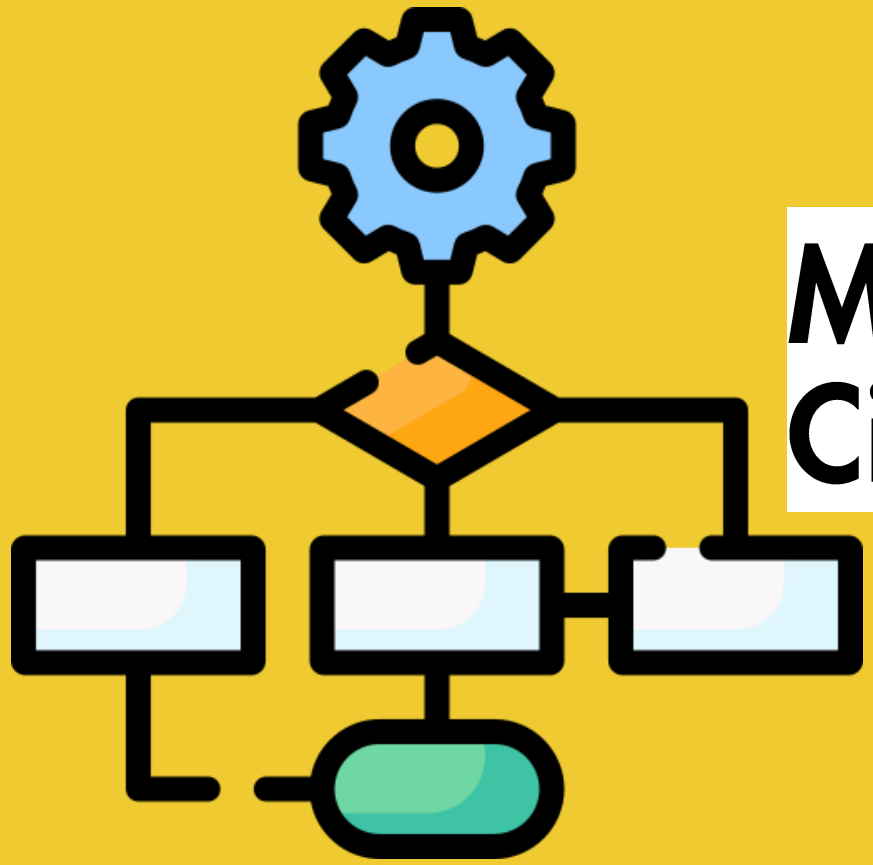
- **Descrição:** Quando o software atinge o fim de sua vida útil, ele é retirado de serviço. Isso pode acontecer quando ele se torna obsoleto, é substituído por uma nova versão ou simplesmente não é mais necessário.
- **Atividades:** Notificação aos usuários, desativação do sistema, migração de dados, arquivamento da documentação.

Exercício 1

Resposta: C

Qual das fases abaixo não faz parte do ciclo de vida do software?

A) Planejamento e Análise de Requisitos
B) Desenvolvimento (Implementação)
C) Marketing e Vendas
D) Testes (Validação e Verificação)
E) Manutenção e Suporte



Modelos de Ciclo de Vida do Software

Modelos de Processo de Software

- Às vezes chamado de ciclo de vida do desenvolvimento de software, ou modelo SDLC (*Software Development Life Cycle*).
- É uma representação simplificada de um processo de software.

Modelos de Ciclo de Vida do Software

- São metodologias que descrevem as fases e atividades necessárias para o desenvolvimento de software, desde a concepção inicial até a entrega e manutenção do produto final.
- Esses modelos servem como guias para gerenciar o processo de desenvolvimento, garantir qualidade e prever prazos e custos.

Modelos de Ciclo de Vida do Software

- O **ciclo de vida do software** pode ser estruturado de diferentes maneiras, dependendo do **modelo** de desenvolvimento adotado.

- Cada modelo representa um processo a partir de uma perspectiva particular e, desse modo, fornece apenas **informações parciais** sobre esse processo.
- Por exemplo, um modelo de atividades do processo **mostra as atividades e sua sequência, mas não os papéis** das pessoas envolvidas nele.

Vamos (re?) apresentar, inicialmente, 3 modelos “clássicos”.

Esses processos são tão genéricos, que as vezes são chamados de Paradigmas de Processo.

Modelos de Processo de Software

Ian Sommerville - Engenharia de Software – Capítulo 2

1. Modelo em **Cascata** (Sequencial, Linear) – modelo clássico até a década de 90.
2. Desenvolvimento **Incremental**.
3. Integração e configuração.

Exercício 2

Resposta: D

Quando os requisitos estão bem definidos, são razoavelmente estáveis e bem compreendidos; quando o trabalho flui de forma linear e rígida, sem retornos; esse modelo de processo sugere uma abordagem sequencial e sistemática para o desenvolvimento de software. Começando com o levantamento de necessidades por parte do cliente, avançando pelas fases de planejamento, modelagem, construção, etc., culminando no suporte contínuo do software concluído".

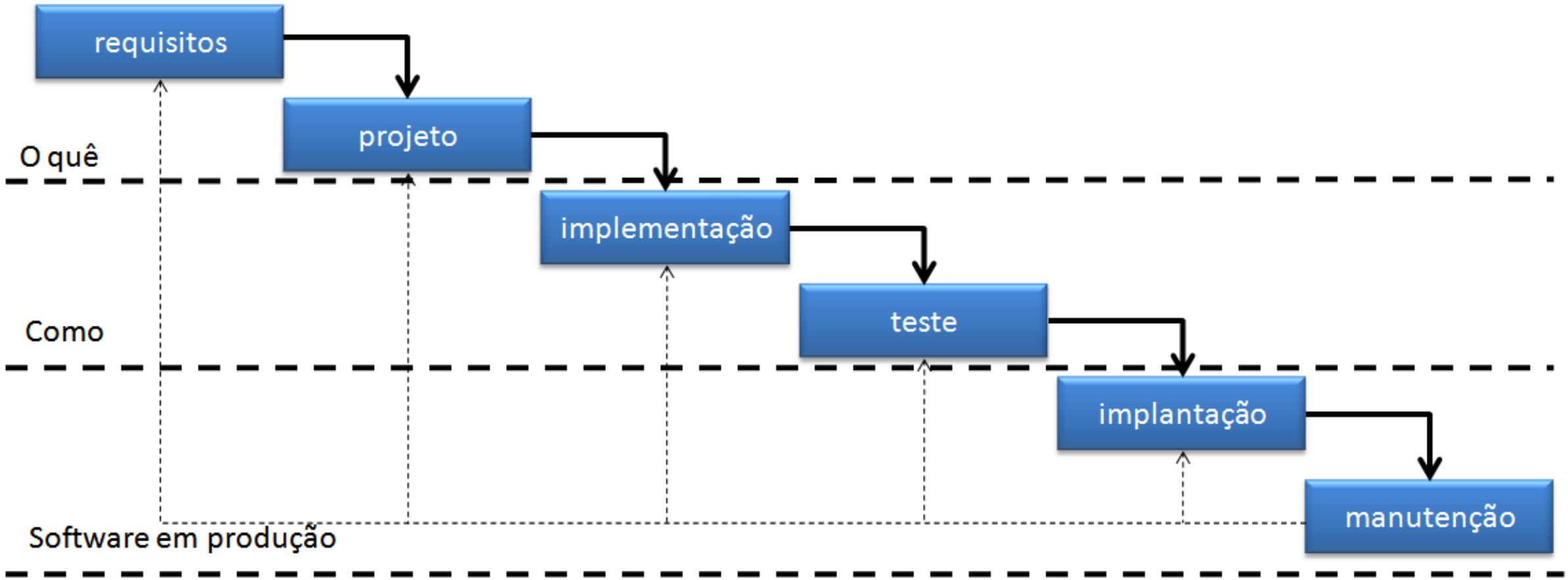
Qual modelo de processo de software está sendo descrito acima?

a) Incremental
b) Evolucionário
c) Espiral
d) Cascata
e) Iterativo

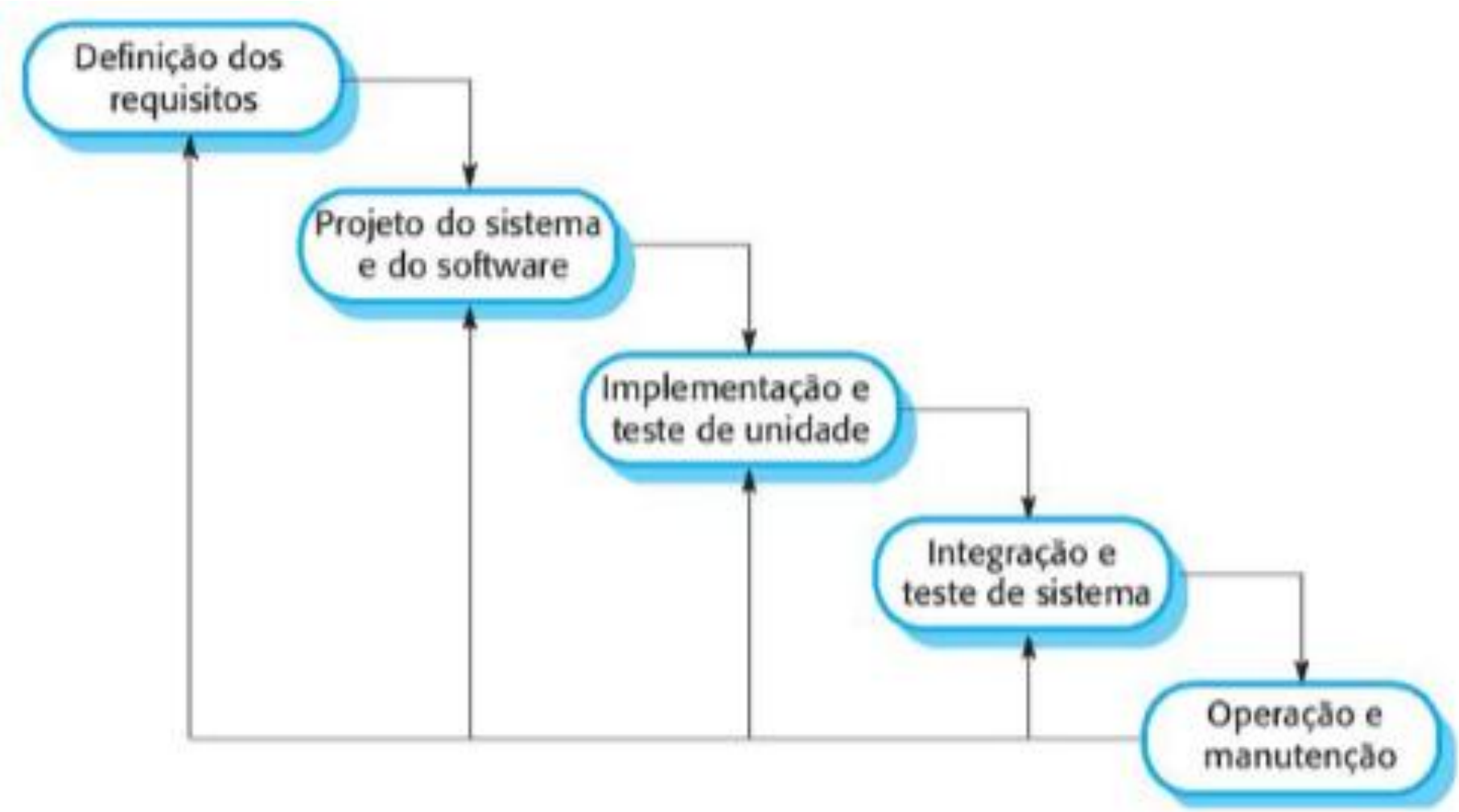
1. MODELO EM CASCATA (WATERFALL)

Modelo em Cascata

Este modelo aborda o desenvolvimento em etapas, sendo estas:



SOMMERVILLE, Ian. Engenharia de software. 10. ed. São Paulo, SP: Pearson, 2018.



Modelo em Cascata

O modelo em cascata **pode ser muito melhor do que não optar por modelo algum** (melhor que nada).

Desvantagens

- Apresenta **baixo rendimento** (muito retrabalho).
- Modelo **muito rígido** (dificultando mudanças posteriores).
- **Demora** para o cliente ver alguma **funcionalidade** do programa.
- **Parte da equipe fica ociosa** durante as etapas que não estão sendo desenvolvidas.

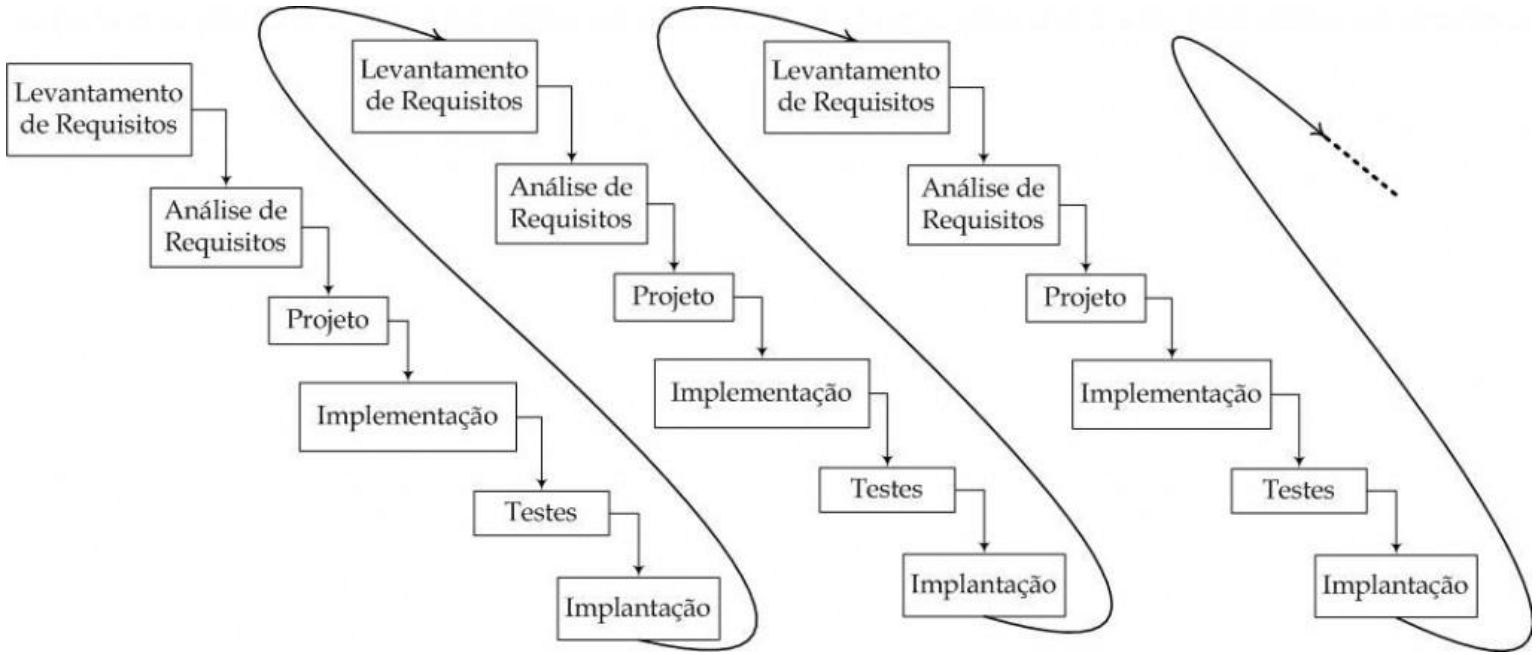
O modelo em cascata é adequado somente para alguns tipos de sistema:

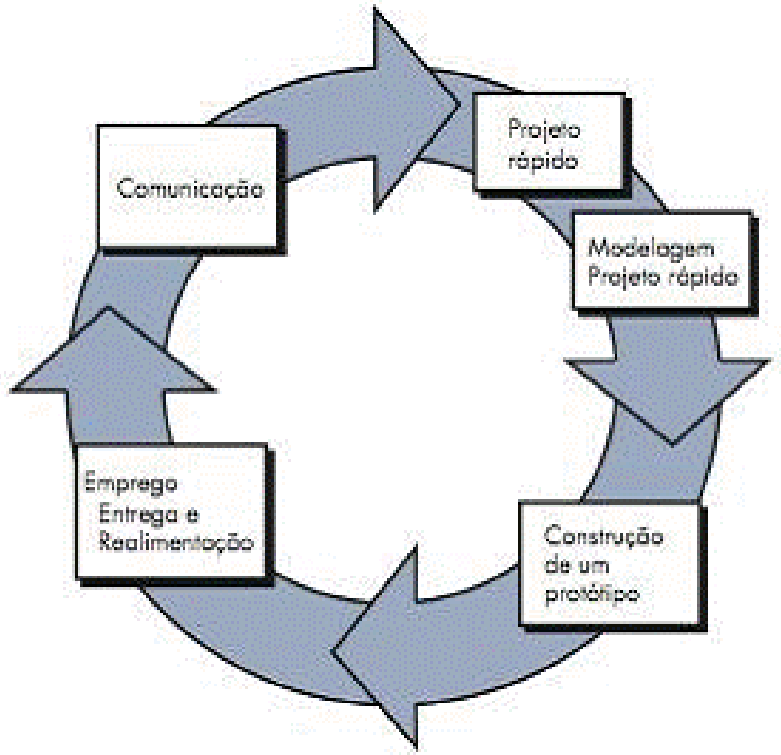
1. **Sistemas embarcados**, nos quais o software deve interagir com sistemas de hardware. Em virtude da inflexibilidade do hardware, normalmente não é possível postergar as decisões sobre a funcionalidade do software até que ele seja implementado.
2. **Sistemas críticos**, nos quais há necessidade de ampla análise da segurança (safety) e segurança da informação (security) da especificação e do projeto do software. Nesses sistemas, os documentos de especificação e de projeto devem estar completos para que a análise seja possível.
3. **Grandes sistemas de software**, que fazem parte de sistemas de engenharia mais amplos, desenvolvidos por várias empresas parceiras.

Mega construções – Aeroporto de Hong Kong. 38 a 43m

https://www.youtube.com/watch?v=tGpXUV2qBWE

2. Desenvolvimento Incremental

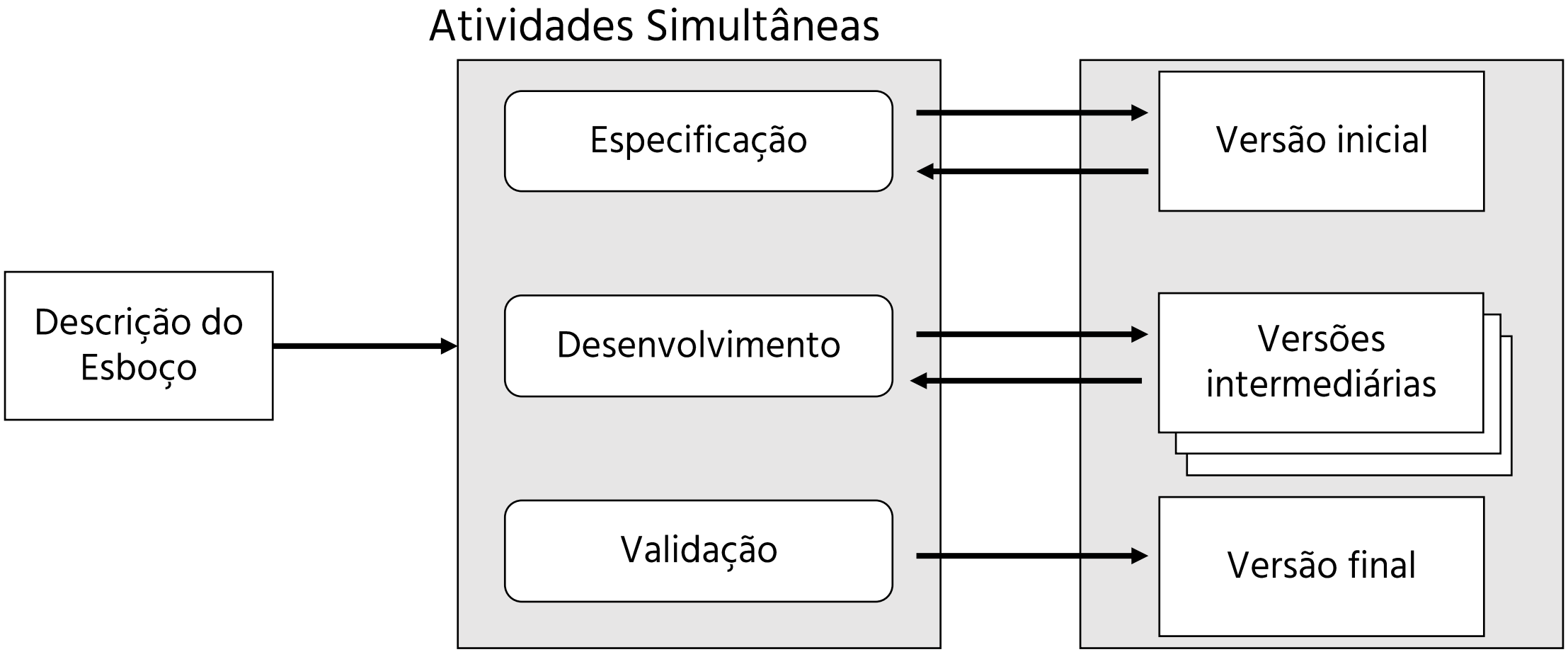




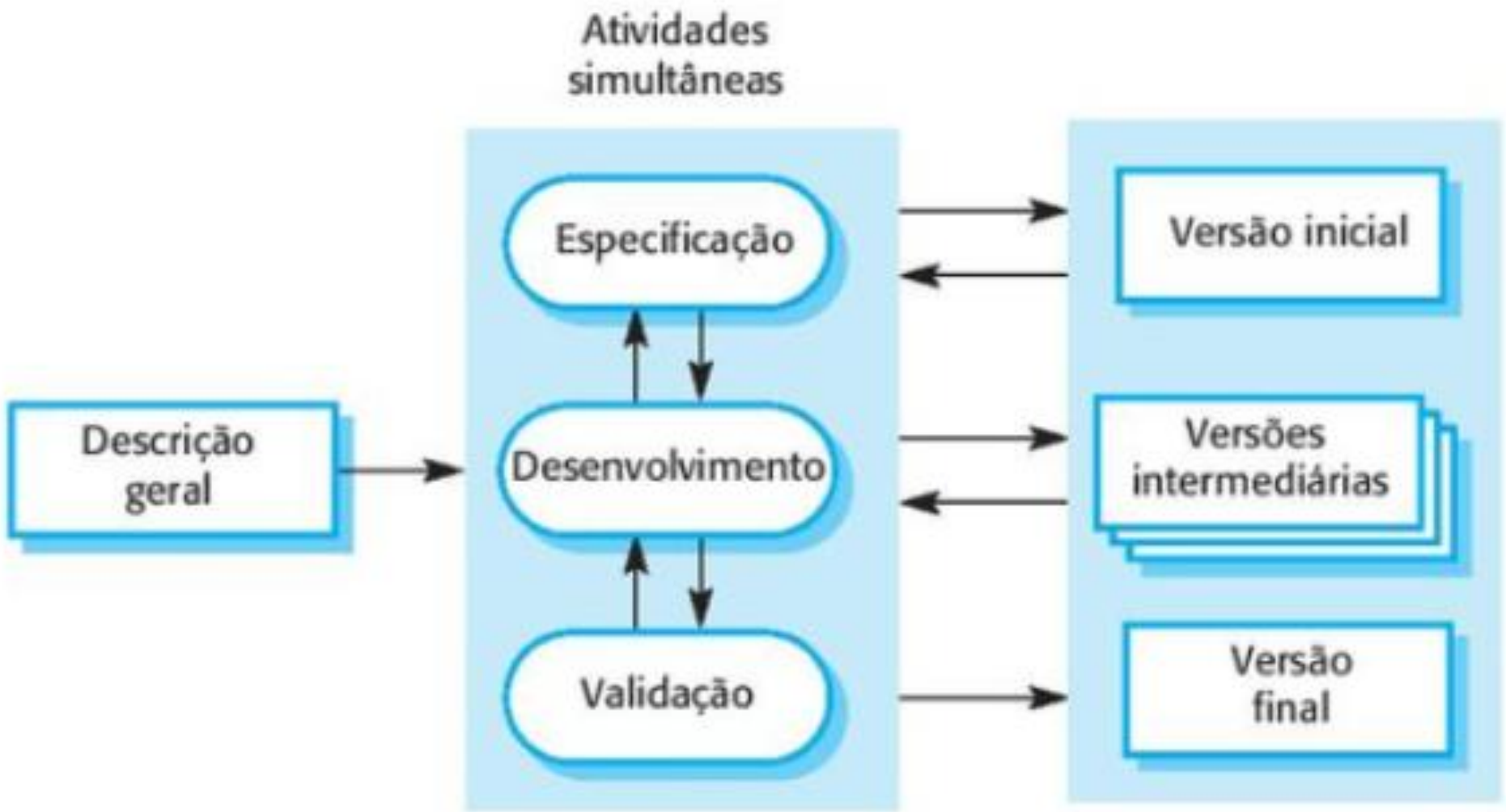
2. Desenvolvimento Incremental

- Baseia-se na ideia de desenvolver uma implementação inicial, expô-la aos comentários do usuário e continuar por meio da criação de várias versões até que um sistema adequado seja desenvolvido.
- As atividades de **especificação**, **desenvolvimento** e **validação** são intercaladas, e não separadas, com **rápido feedback** entre todas as atividades.

Desenvolvimento Incremental



SOMMERVILLE, Ian. Engenharia de software. 10. ed. São Paulo, SP: Pearson, 2018.



2. Desenvolvimento Incremental

- É **melhor que** uma abordagem **em cascata** para a maioria dos sistemas de negócios **cujos requisitos** estão propensos a mudar **durante a fase de desenvolvimento**.
- Reflete a forma que resolvemos problemas: raramente desenvolvemos uma solução completa. Damos um passo em direção a solução e recuamos quando percebemos um erro.
- Cada **incremento incorpora uma funcionalidade** necessária para o cliente (em geral começamos com as maiores necessidades).

Vantagens do Desenvolvimento Incremental em Relação ao Modelo em Cascata

1. O **custo de acomodar as mudanças** nos requisitos do cliente é reduzido.
2. É **mais fácil obter feedback** dos clientes sobre o desenvolvimento que foi feito.
3. É possível obter entrega e **implementação rápida de um software útil** ao cliente, mesmo se toda a funcionalidade não for incluída.

Desvantagens do Desenvolvimento Incremental

Do ponto de vista do **gerenciamento**, a abordagem incremental tem dois problemas:

1. **O processo não é visível.** A produção de documentos de cada incremento pode se tornar custosa.
2. **A estrutura do sistema tende a degradar** com a adição de novos incrementos. Refatorar a cada adição demanda tempo e dinheiro.

2. Desenvolvimento Incremental

Os **problemas** do desenvolvimento incremental são **particularmente críticos para os sistemas de vida-longa grandes e complexos**, nos quais várias equipes desenvolvem diferentes partes do sistema.

3. Integração e Configuração

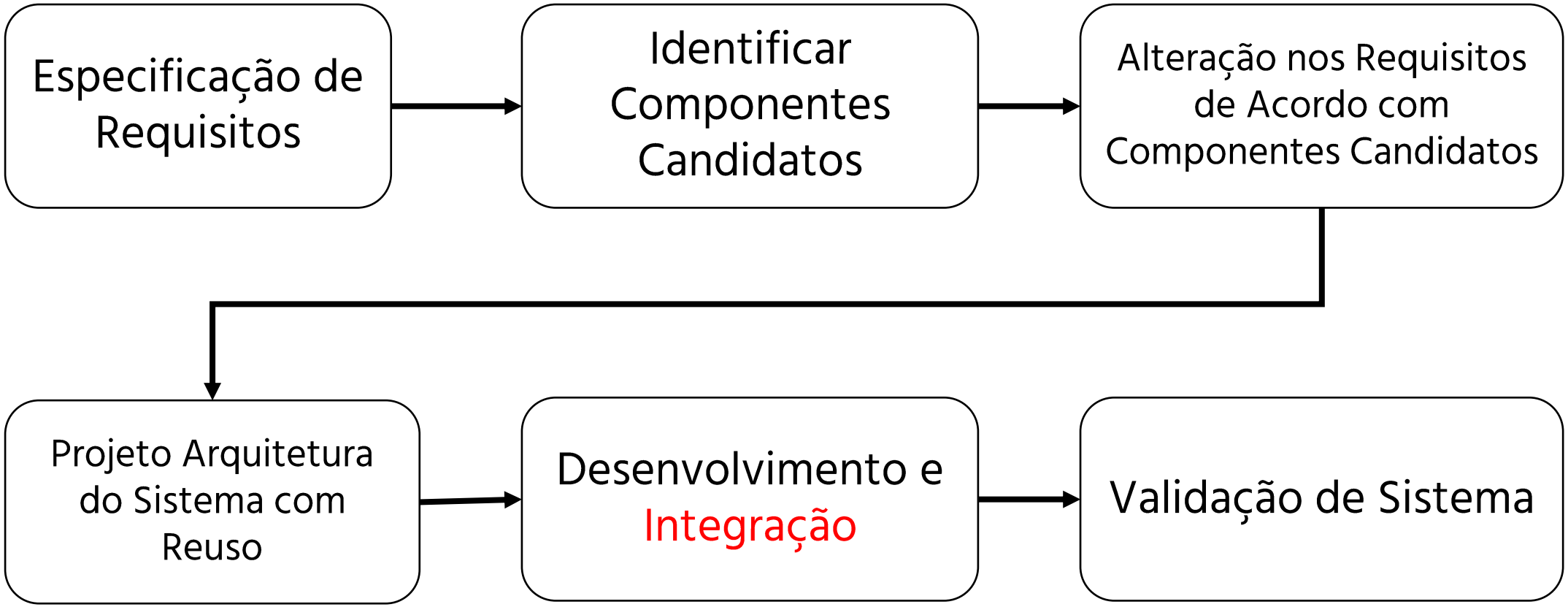
3. Integração e Configuração

- Na maioria dos projetos, existe o reuso de software.
- No século XXI esse tipo de prática tornou-se comum e **depende de uma ampla base de componentes reusáveis** de software de um framework.
- Os estágios iniciais de desenvolvimento são comparáveis a outros processos, mas os estágios internos são diferentes.
- As abordagens orientadas ao reuso **contam com bases de componentes de software reusáveis e um framework de integração** para a composição desses componentes.

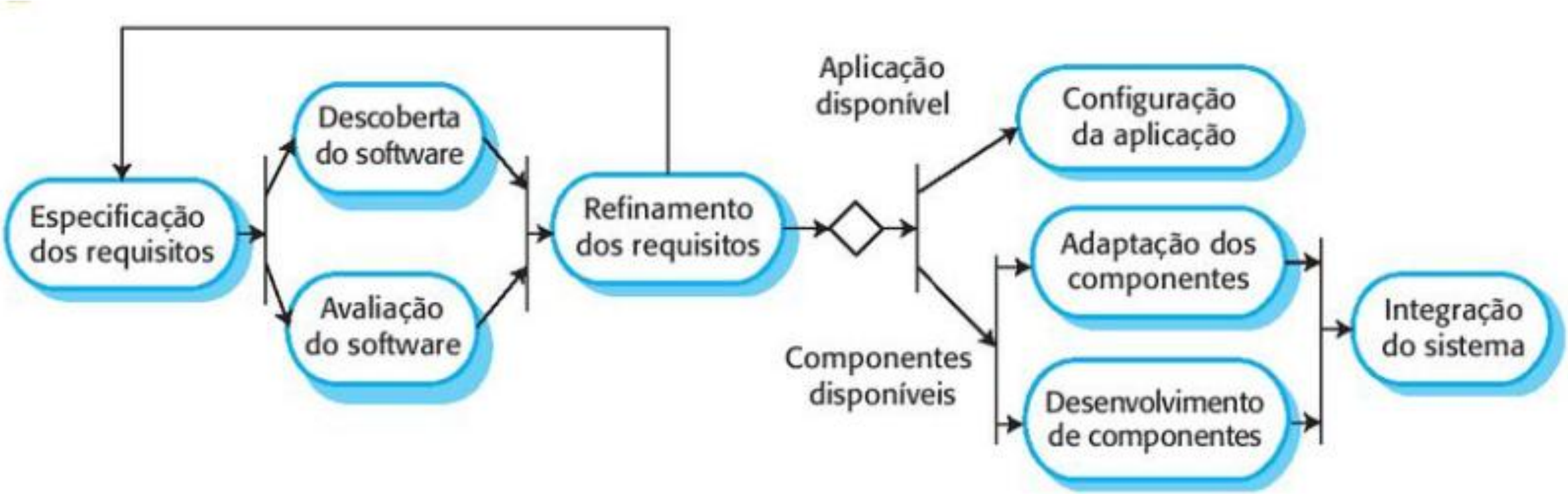
Três tipos de componentes de software são reusados frequentemente

1. **Sistemas de aplicação stand-alone** (não dependem de outros sistemas, ex: processador de texto, app desktop) configurados para utilização em um ambiente particular. Esses sistemas são de uso geral e possuem muitas características, mas precisam ser adaptados para uso em uma aplicação específica.
2. **Coleções de objetos desenvolvidos como um componente ou como um pacote** a ser integrado a um framework de componentes, como o Java Spring framework (WHEELER; WHITE, 2013).
3. **Web services** desenvolvidos de acordo com os padrões de serviço e que estão disponíveis para uso remoto na internet.

Em Sommerville (2011) esse modelo era chamado de Engenharia de Software Orientada a Reuso



SOMMERVILLE, Ian. Engenharia de software. 10. ed. São Paulo, SP: Pearson, 2018.



Vantagens da Integração e Configuração

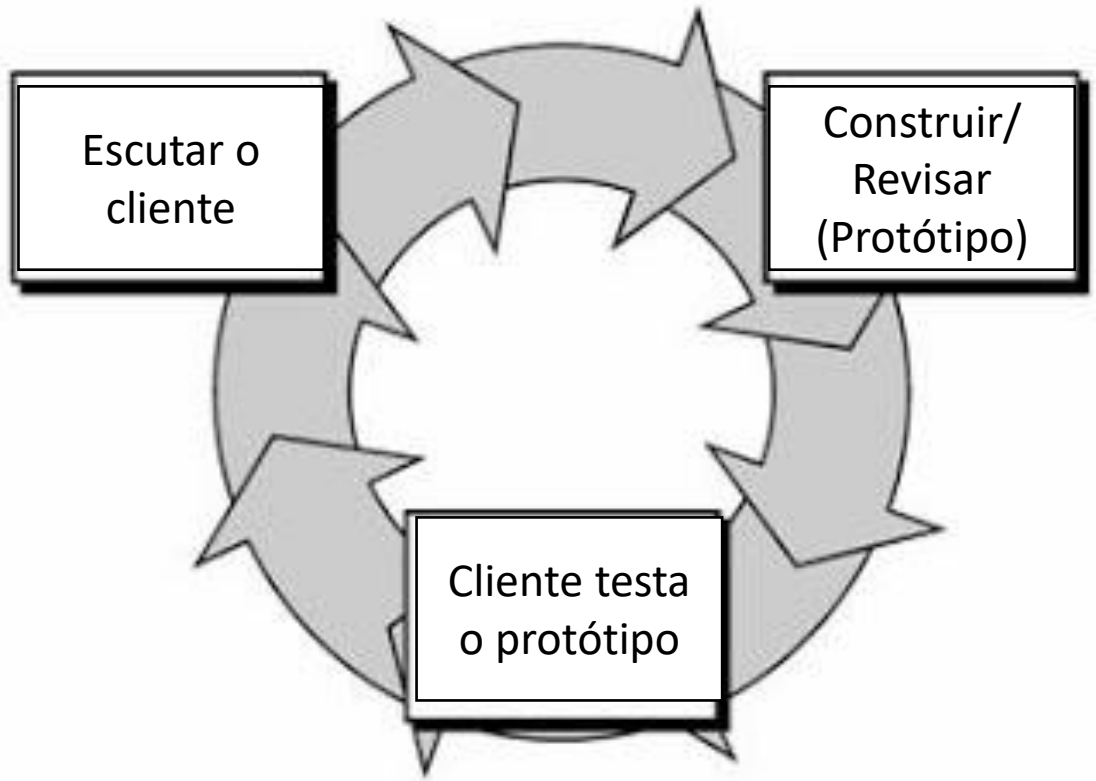
- Reduz a quantidade de software a ser desenvolvido, reduzindo também custos e riscos.
- Geralmente proporciona entrega mais rápida.

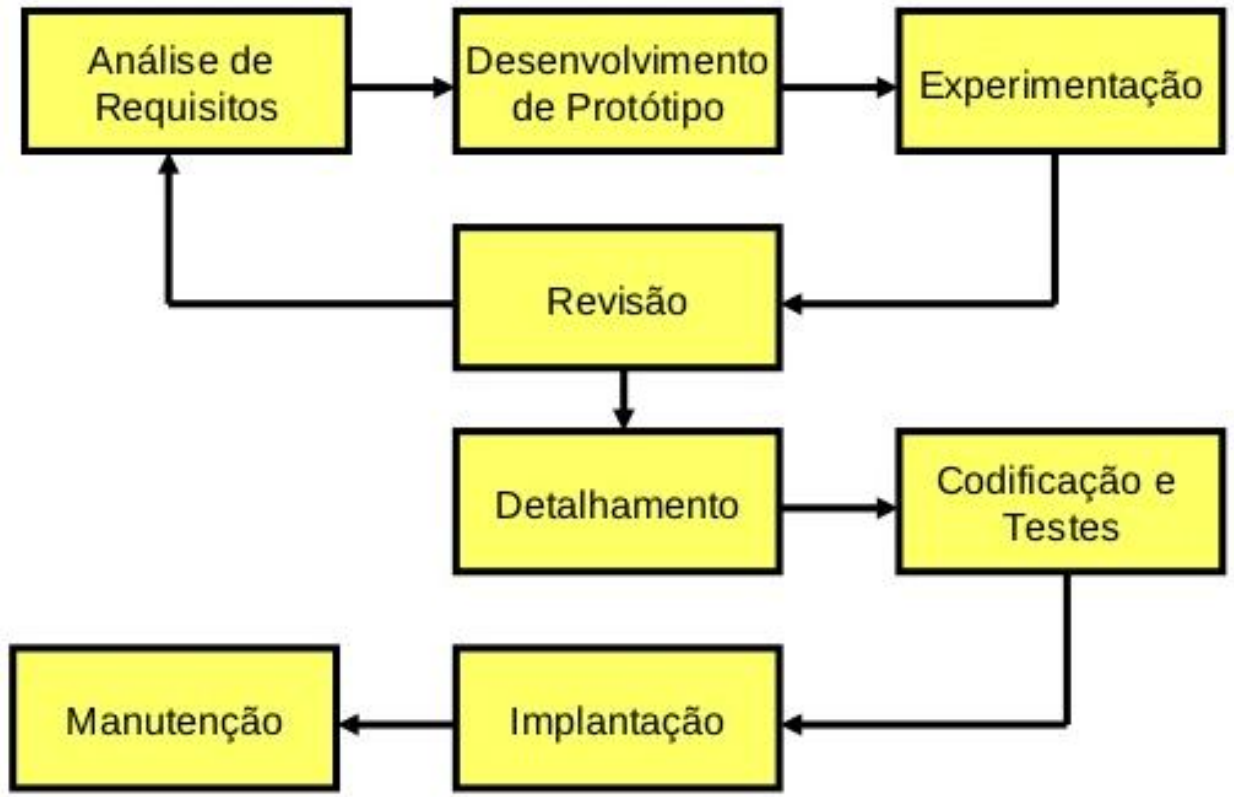

ENTRE OS MODELOS APRESENTADOS, QUAL O MELHOR?

Intervalo: 15 minutos

Outros Modelos

Prototipação





Prototipação

- É baseado no **desenvolvimento de um protótipo** com base no conhecimento dos requisitos iniciais do sistema.
- Este modelo propõe um ciclo de atividades que se repetem até concluir o projeto:
  - Coleta de Requisitos
  - Projeto Rápido
  - **Construção** do Protótipo
  - Avaliação
  - Refinamento

Prototipação

- O modelo prototipação permite maior estudo dos requisitos e diminui a chance de retrabalho.
- Porém... Fazer muitos protótipos pode acabar levando muito tempo além de encarecer o projeto.

Desvantagens

+ Tempo

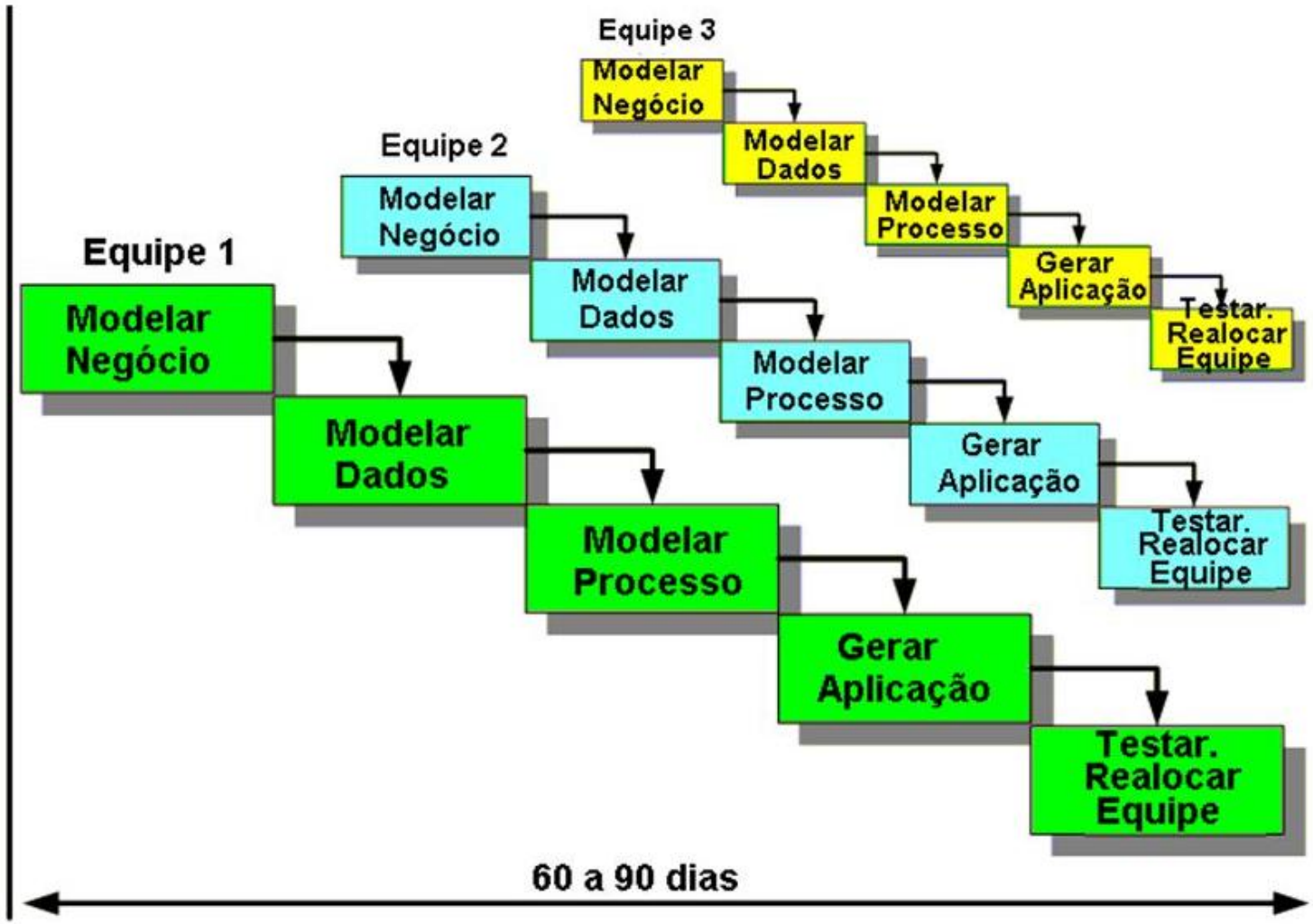
+ Dinheiro


DUNGEONS & DRAGONS
QUAL A DIFERENÇA ENTRE PROTOTIPAÇÃO E O DESENVOLVIMENTO INCREMENTAL?
ESTA É FÁCIL MESTRE! É O DESCARTE DO PROTÓTIPO DA PROTOTIPAÇÃO!

RAD - *Rapid Application Development*

- Foca em rapidez dividindo o desenvolvimento em pequenos processos em cascatas a serem **executados por várias equipes**.
- No entanto, pode existir muitas **incompatibilidades** entre as partes desenvolvidas por cada equipe.
- Funciona melhor em empresas com maior maturidade, bem padronizada e com funcionários experientes.

RAD
Rapid Application Development

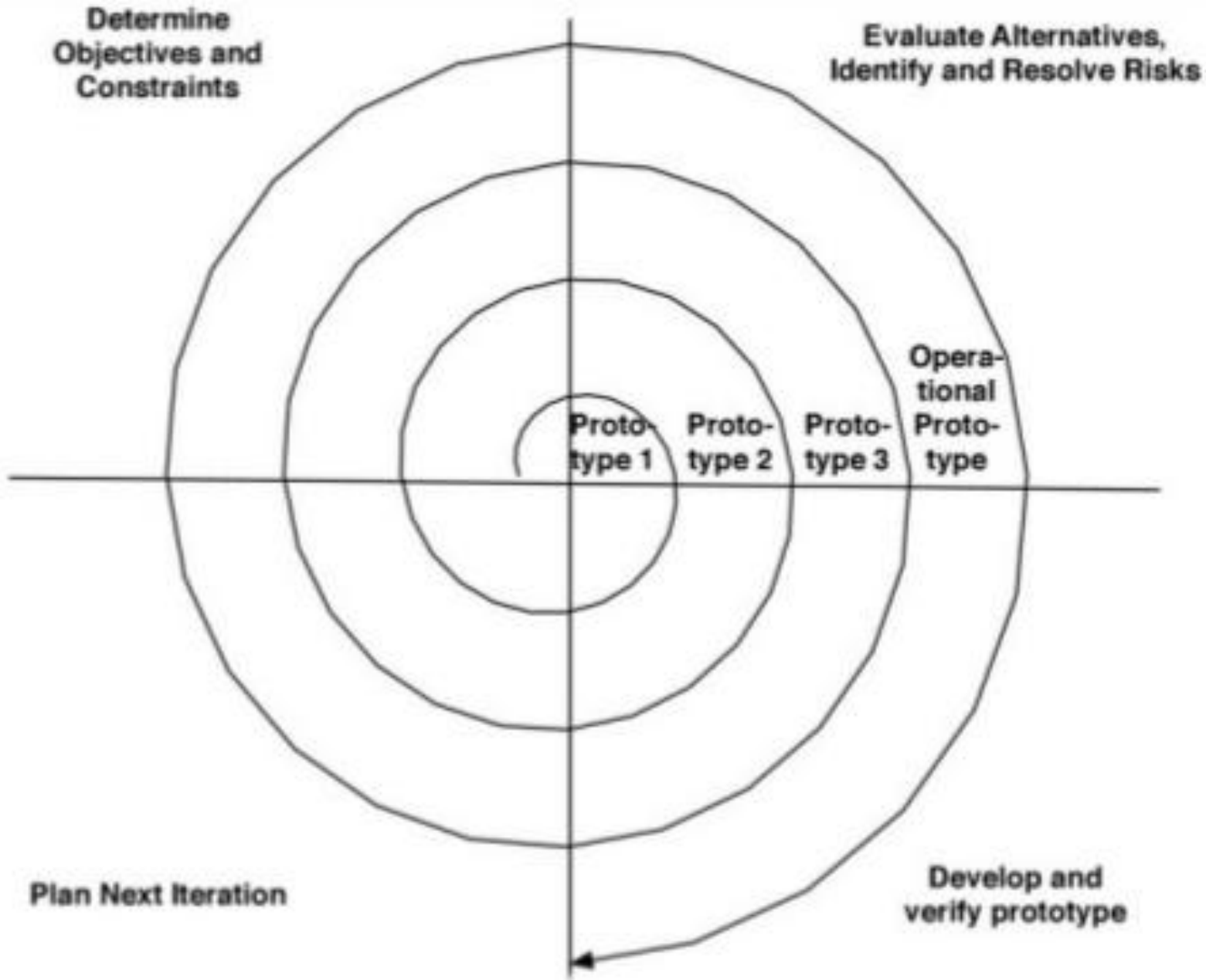


RAD
Rapid Application Development

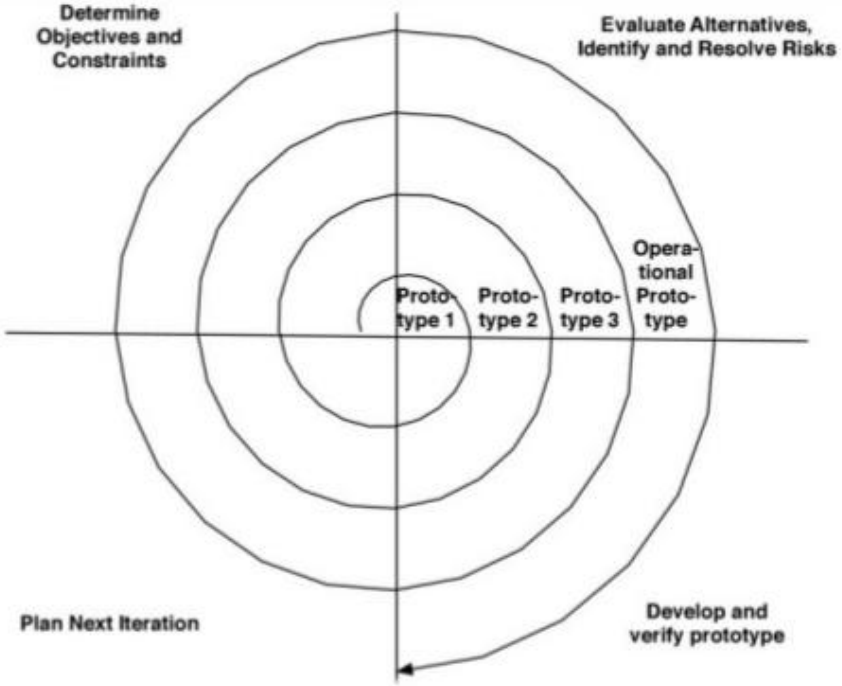
Desvantagem

Seu maior problema é a **dificuldade em juntar** o trabalho final.

Espiral

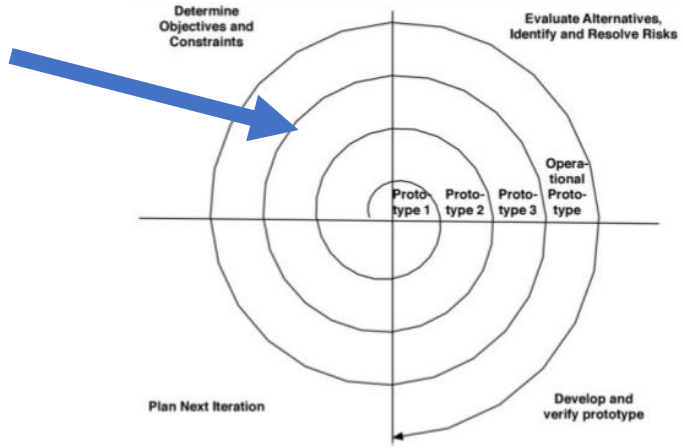

Determine Objectives and Constraints
Evaluate Alternatives, Identify and Resolve Risks
Proto-type 1
Proto-type 2
Proto-type 3
Opera-tional Proto-type
Plan Next Iteration
Develop and verify prototype

Espiral



- Baseia-se em prover um **metamodelo** que pode acomodar diversos processos específicos, ou seja...
- Podemos **encaixar** nele as **principais características dos modelos vistos anteriormente**, adaptando-os a necessidades específicas de desenvolvedores ou particularidades do software a ser desenvolvido.
- O diferencial desse modelo é o seu **reconhecimento de riscos**.

Espiral



- O processo é representado por uma espiral onde **cada quadrante significa**:
  I. A definição dos objetivos,
  II. Avaliação e redução dos riscos,
  III. Escolha da abordagem do desenvolvimento,
  IV. Validação e planejamento;
- No planejamento é decidido a continuidade do projeto e, caso decida-se pela continuidade, as próximas fases do projeto serão elaboradas.

Pergunta!

Resposta: D

Considere as asserções a seguir feitas sobre um modelo de processo de software.

I. Combina a natureza iterativa de modelos incrementais com aspectos sistemáticos do modelo em cascata.

II. Pode ser aplicado em todo ciclo de vida de uma aplicação, inclusive, após a entrega do software.

III. É um modelo que reconhece explicitamente a necessidade de gerenciar riscos.

Após analisar as asserções categorize a qual modelo de processo de software elas pertencem.

a) Modelo clássico b) Protótipo evolucionário c) Processo unificado

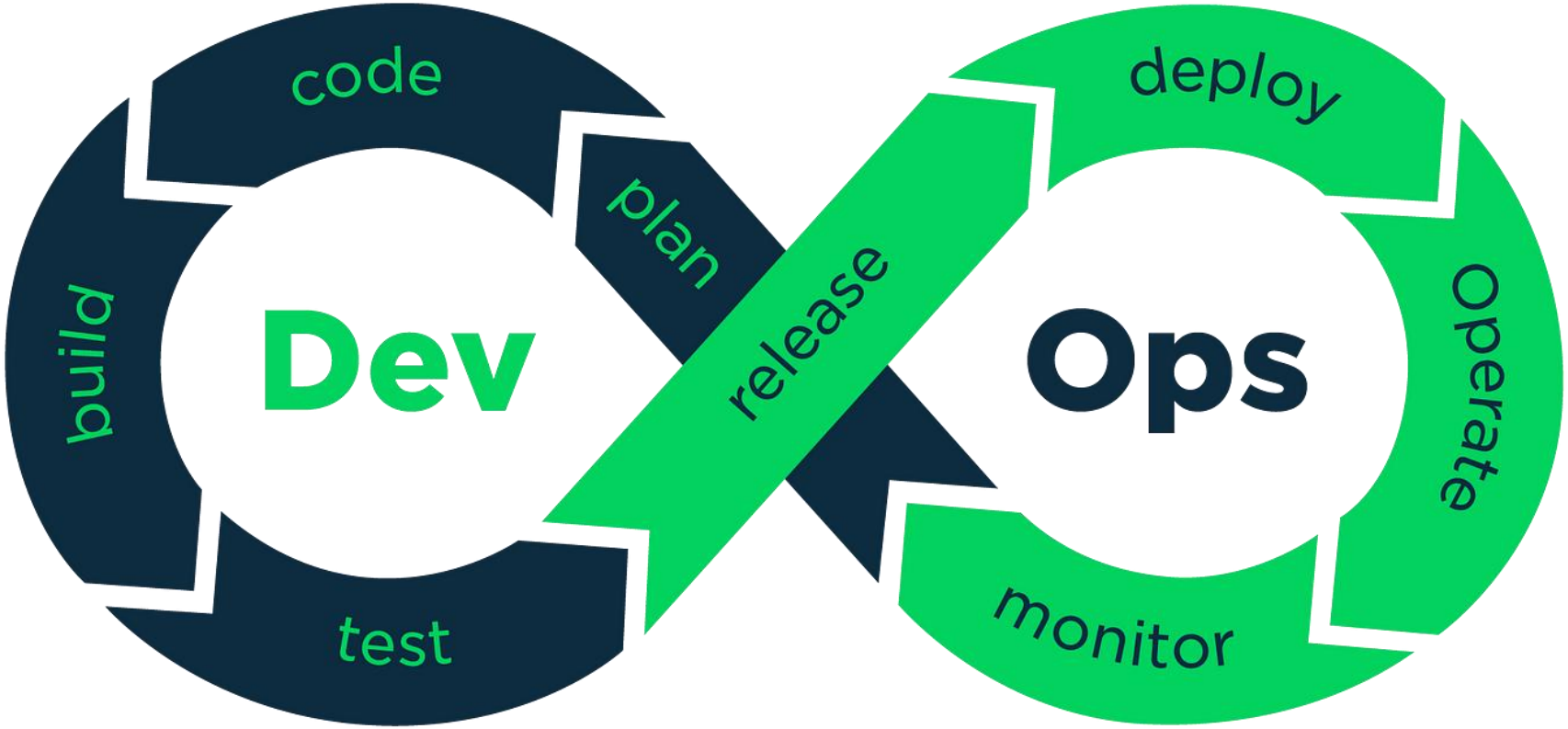
d) Modelo espiral e) Modelo de métodos formais

Outros modelos...

Modelo DevOps

- **Descrição:** Integra desenvolvimento (Dev) e operações (Ops) para melhorar a colaboração entre equipes e acelerar o ciclo de entrega do software.
- **Características:** Foco na automação, integração contínua e entrega contínua (CI/CD), e monitoramento constante.
- **Vantagens:** Reduz o tempo de lançamento no mercado, melhora a colaboração entre equipes, aumenta a qualidade e a confiabilidade.
- **Desvantagens:** Exige uma mudança cultural significativa e pode ser difícil de implementar em ambientes tradicionais.

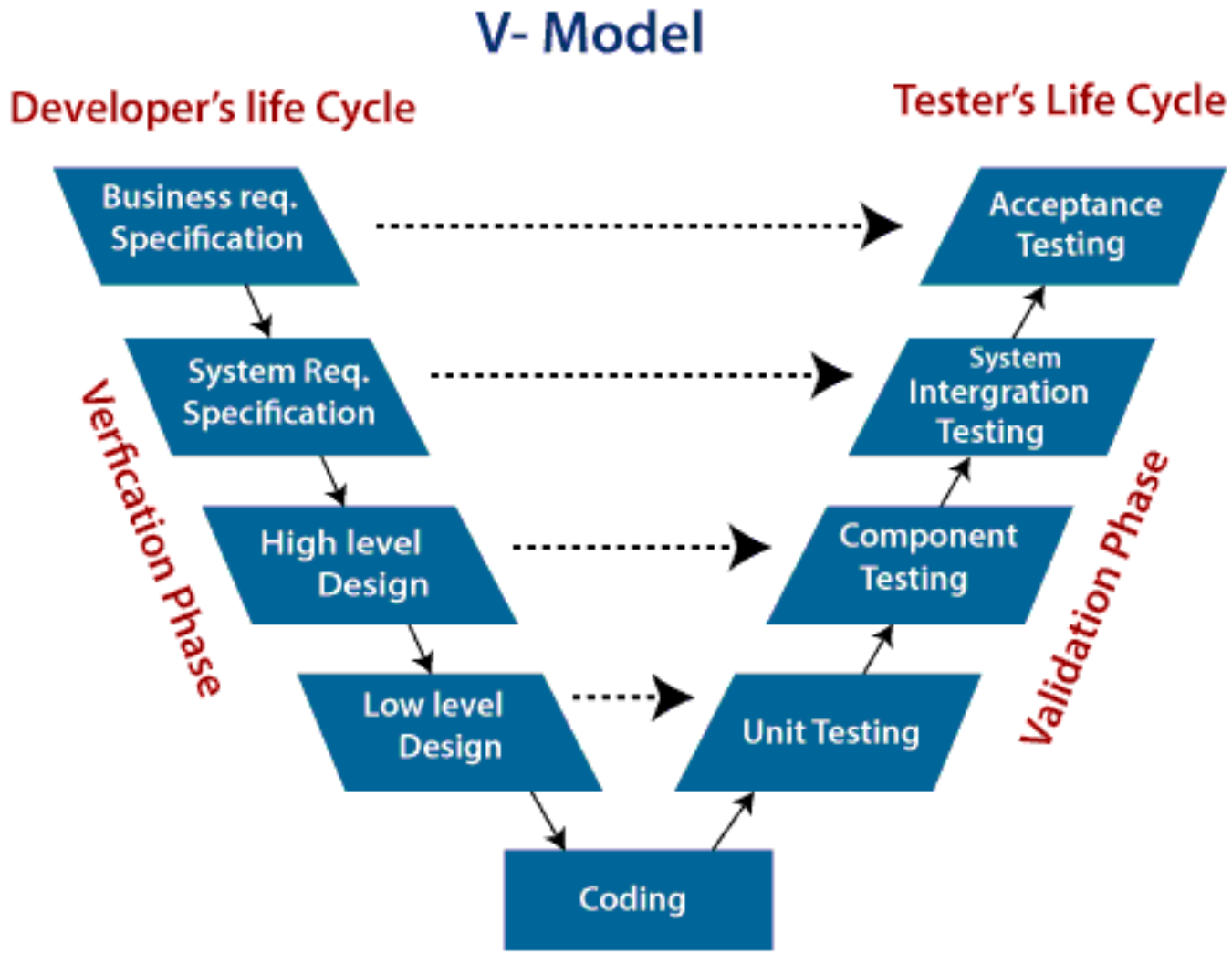
Modelo DevOps



Modelo V-Shape

- **Descrição:** Similar ao modelo cascata, mas com a diferença de que cada fase de desenvolvimento tem uma fase correspondente de teste, formando uma espécie de “V” na representação gráfica.
- **Características:** Foco na verificação e validação; cada fase deve ser concluída antes que a seguinte comece.
- **Vantagens:** Melhora o controle e a qualidade através de testes planejados desde o início.
- **Desvantagens:** Inflexível para mudanças, adequado principalmente para projetos com requisitos estáveis.

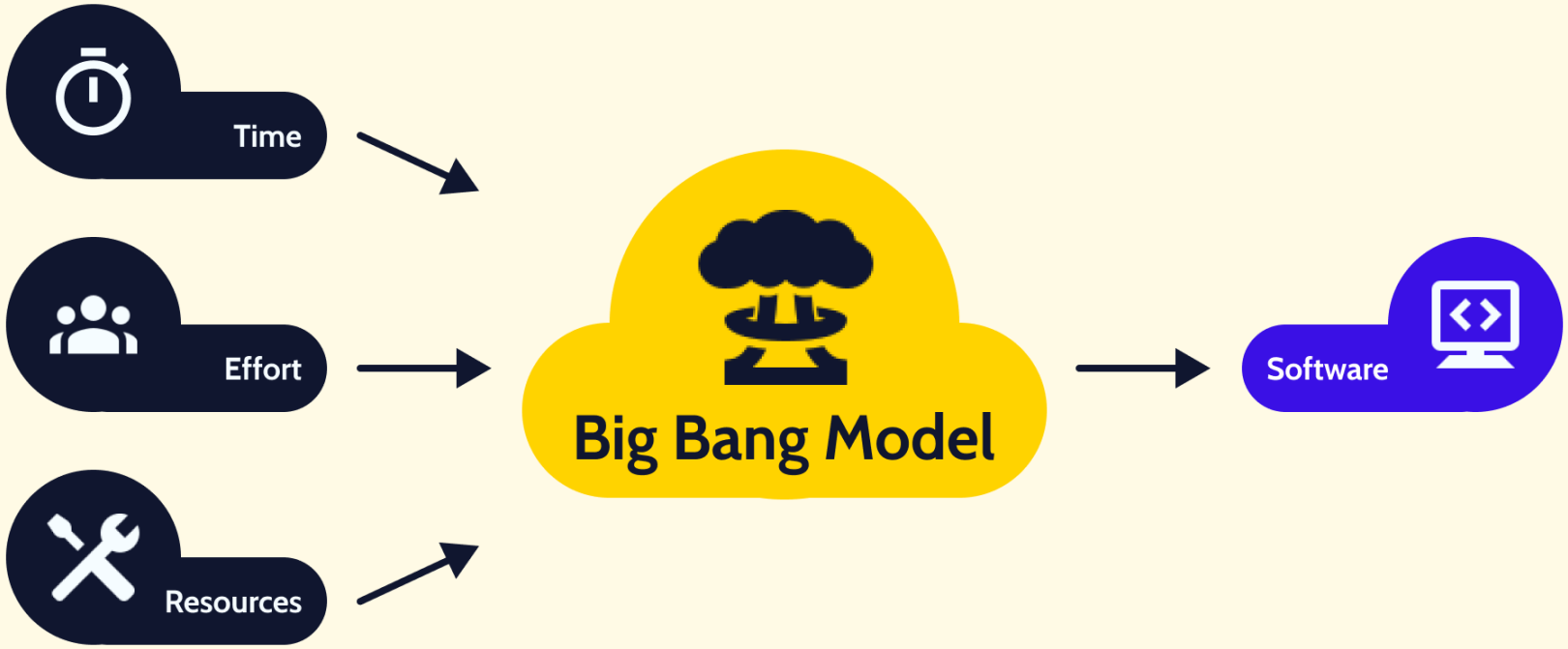
Modelo V-Shape



Modelo Big Bang

- **Descrição:** Um modelo ad-hoc, onde os desenvolvedores começam a codificar com poucos ou nenhum requisito bem definido.
- **Características:** Sem planejamento formal, foco na entrega rápida.
- **Vantagens:** Simplicidade, adequação para pequenos projetos com poucos desenvolvedores.
- **Desvantagens:** Alto risco de falhas, difícil de gerenciar e manter, não adequado para projetos complexos.

Modelo Big Bang



LIÇÕES
PARA
ENGENHARIA DE SOFTWARE DO PEQUENO PRÍNCIPE

Então desenhei.

Ele olhou atentamente e disse:

— Não! Este já está muito doente. Faça um outro.

Desenhei:

Meu amigo sorriu gentilmente, com indulgência:

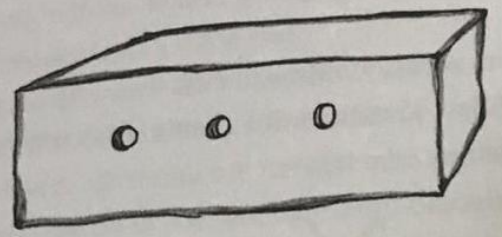
— Olha... este não é um carneirinho, é um carneiro adulto. Ele já tem chifres...

Fiz de novo meu desenho:

Mas ele foi recusado, como os anteriores.

— Este está muito velho. Quero um carneirinho que viva bastante.

Então, sem paciência, como estava com pressa para começar a desmontagem do meu motor, rabisquei este desenho aqui:



E arrisquei:

— Esta é a caixa. O carneirinho que você quer está aí dentro.

Mas fiquei muito surpreso em ver iluminar-se o rosto do meu jovem juiz:

— É assim mesmo que eu queria! Você acha que esse carneirinho precisa de muito capim?

— Por quê?

— Porque onde moro é bem pequeno...

— Vai bastar, com certeza. Eu lhe dei um carneirinho bem pequenino.

Ele debruçou-se sobre o desenho:

— Nem tão pequeno assim.... Olha! Ele dormiu...

E foi assim que conheci o pequeno príncipe.

Exercício de Fixação

Considerando os modelos: (i) Cascata, (ii) Incremental, (iii) Engenharia de Software Orientada a Reuso, (iv) Prototipação, (v) RAD e (vi) Espiral...

1 - Faça um prompt em alguma IA Generativa para criar um quadro contendo as **vantagens** e **desvantagens** de cada modelo de processo de software apresentados na aula de hoje.

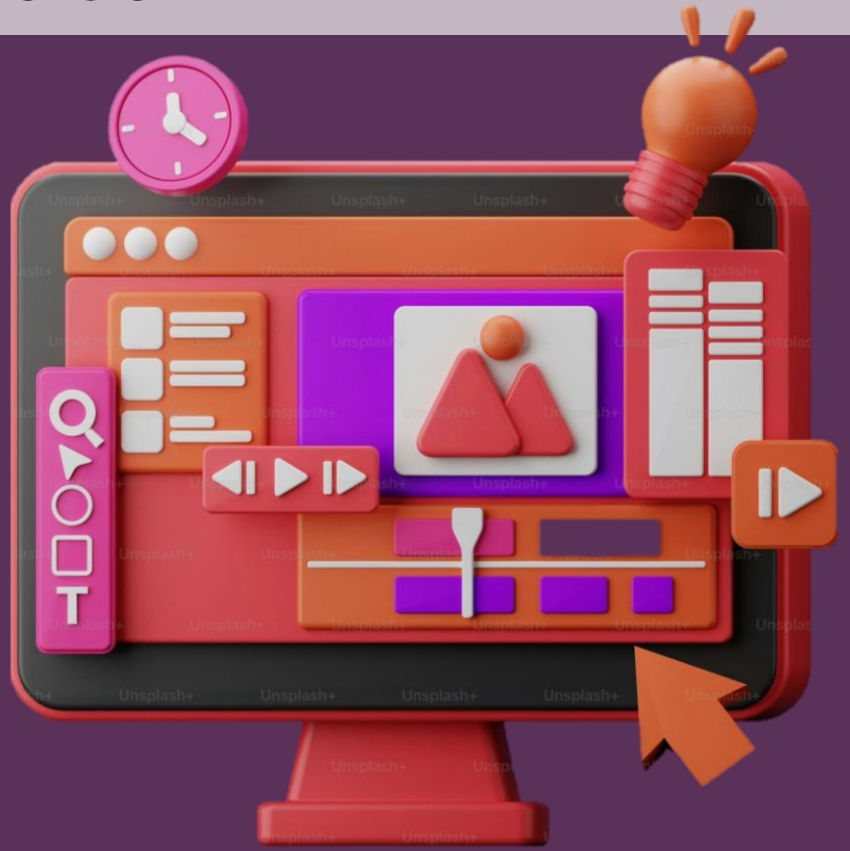
2 – Peça também para **identificar** a **melhor e pior situação** de uso para cada modelo de processo de software.

3 – Compartilhe seu prompt no grupo do WhatsApp da turma.

Lista de Exercícios 2

ENGENHARIA DE SOFTWARE

AULA 2 – MODELOS DE CICLO DE VIDA DO SOFTWARE



WALTER TRAVASSOS SARINHO

@PROF.WALTERTRAVASSOS

WALTER.TRAVASSOS@SECTRAS.EDU.BR