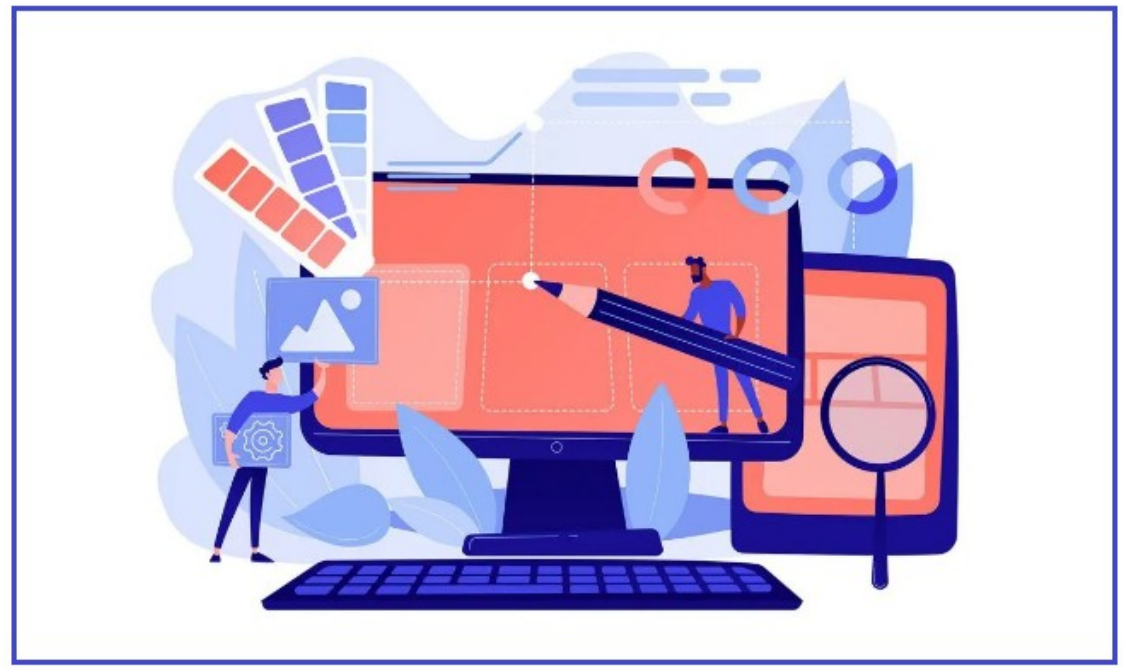
Modelos Métodos e Técnicas de Engenharia de Software

Prof. Sônia A Santana

Ciclo de Vida



Processos de Desenvolvimento de Softwares

Deadlines

- **Inicio 20/02/2024 – Terça-Feira**
- **Avaliações**
  - **1a Avaliação Objetiva (A1) - 08 a 09/05**
    - **24/05 2a oportunidade**
  - **2a Avaliação Discursiva (A2) - 11 a 12/06**
    - **21/06 2a oportunidade**
  - **3a Avaliação Trabalho (A3) - 10/06**
- **Término Regular - 28/06**

Recordando...

- A engenharia de software é uma disciplina da engenharia que se preocupa com todos os aspectos da produção de software.
- Atributos essenciais do produto de software são a manutenibilidade, confiança, proteção, eficiência e aceitabilidade.
- As atividades de alto nível de especificação, desenvolvimento, validação e evolução fazem parte de todos os processos de software.
- As ideias fundamentais da engenharia de software são universalmente aplicáveis a todos os tipos de desenvolvimento do sistema.
- Existem muitos tipos diferentes de sistemas e cada um requer ferramentas de engenharia de software e técnicas apropriadas para o seu desenvolvimento.

Recordando...

Atividade:

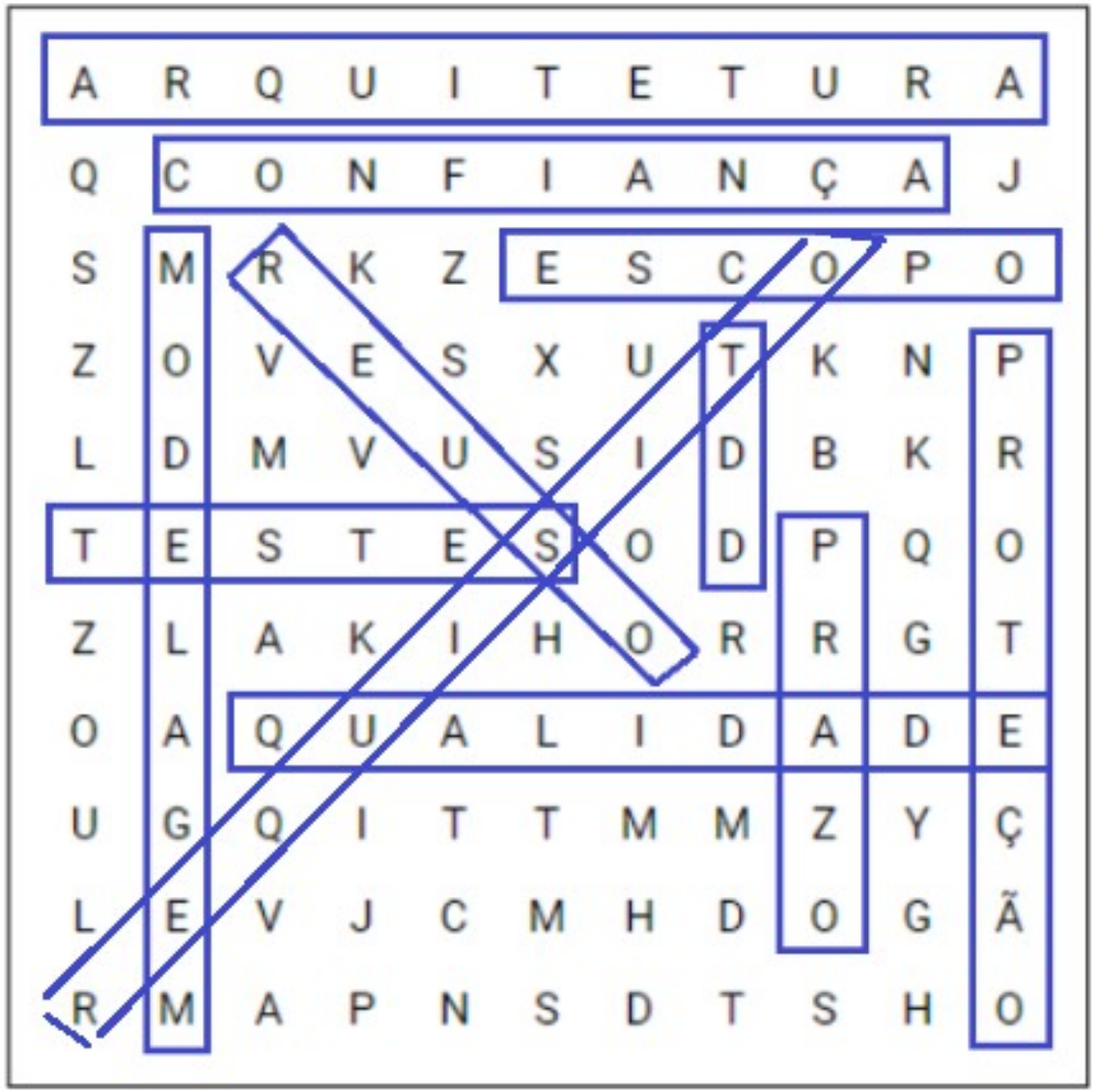
- No Quadro a seguir há 10 palavras que fazem referência aos nossos conteúdos.
- Encontre-as e discuta com seus colegas o significado de cada uma delas dentro do contexto de nossa UC.
(tempo 15 minutos)

Recordando...

- Atividade: Caça palavras

A	R	Q	U	I	T	E	T	U	R	A
Q	C	O	N	F	I	A	N	Ç	A	J
S	M	R	K	Z	E	S	C	O	P	O
Z	O	V	E	S	X	U	T	K	N	P
L	D	M	V	U	S	I	D	B	K	R
T	E	S	T	E	S	O	D	P	Q	O
Z	L	A	K	I	H	O	R	R	G	T
O	A	Q	U	A	L	I	D	A	D	E
U	G	Q	I	T	T	M	M	Z	Y	Ç
L	E	V	J	C	M	H	D	O	G	Ã
R	M	A	P	N	S	D	T	S	H	O

Solução


A R Q U I T E T U R A
Q C O N F I A N Ç A J
S M R K Z E S C O P O
Z O V E S X U T K N P
L D M V U S I D B K R
T E S T E S O D P Q O
Z L A K I H O R R G T
O A Q U A L I D A D E
U G Q I T T M M Z Y Ç
L E V J C M H D O G Ã
R M A P N S D T S H O

Criar Software .... É difícil!

Criar Software …. É difícil!

- Dados do Chaos Report
  - 10% dos projetos terminam no prazo
  - 25% dos projetos não chegam ao Fim
  - 60% dos projetos ficam acima do Custo esperado.
  - Atraso médio de entrega: 1 ano

Criar Software... É Difícil!

Como Solucionar?

Tentativas de lidar com a ***complexidade*** e de ***minimizar os problemas*** contidos no desenvolvimento de software envolvem a definição de ***processo no desenvolvimento de software***.

Por que usar Modelos de Processos de Desenvolvimento de Softwares

RESOLUÇÃO TRADICIONAL PARA TODOS OS PROJETOS

	2011	2012	2013	2014	2015
Bem-sucedidos	39%	37%	41%	36%	36%
Fracassados	22%	17%	19%	17%	19%

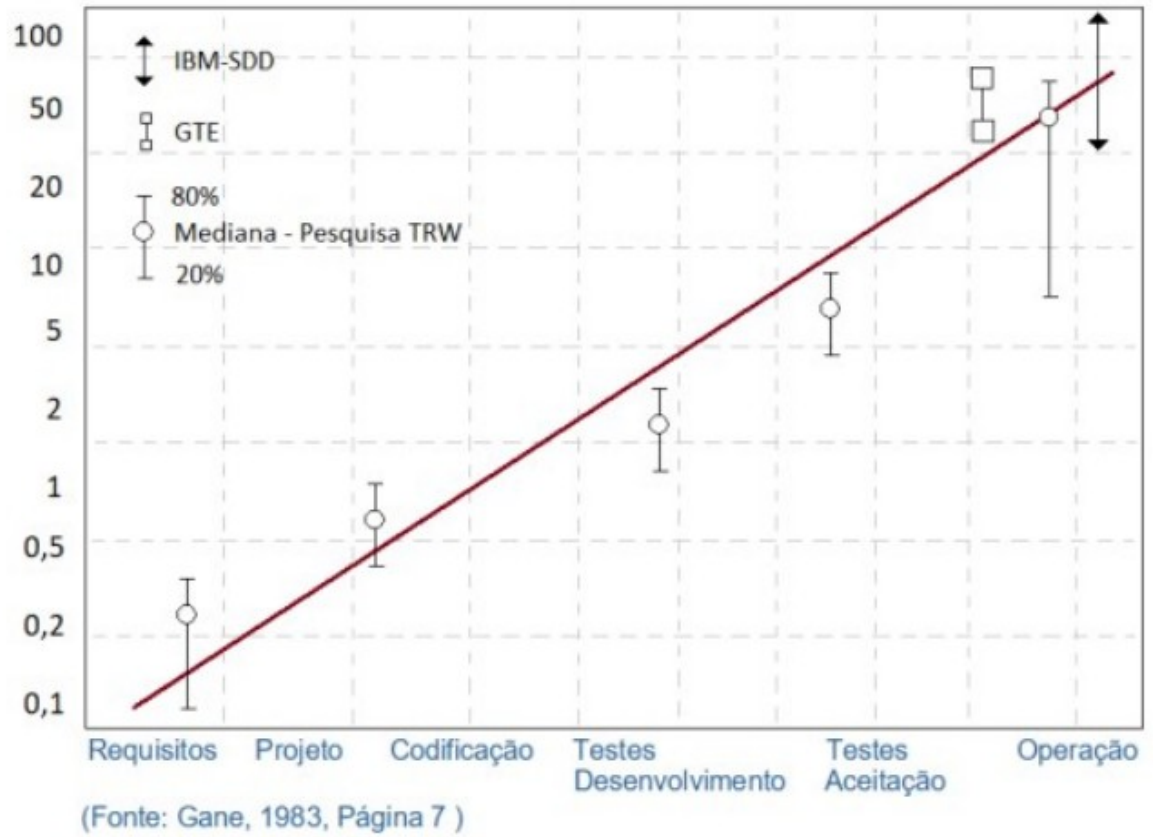
Resolução tradicional de todos os projetos de software. Fonte CHAOS REPORT 2015.

Por que usar Modelos de Processos de Desenvolvimento de Softwares

- Acompanhar o desenvolvimento de software,
- Gerenciar requisitos
- Usar uma arquitetura adequada de desenvolvimento,
- Usar modelos para desenvolver software,
- Controlar continuamente a qualidade do software,
- Acompanhar as mudanças.

Por que usar Modelos de Processos de Desenvolvimento de Softwares

Custo relativo para corrigir um erro durante o desenvolvimento do sistema



(Fonte: Gane, 1983, Página 7)

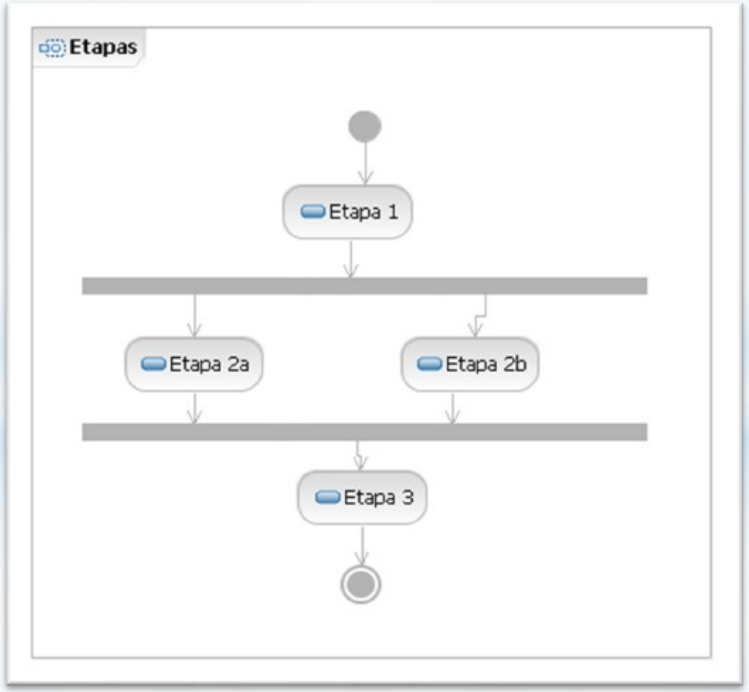
O que é um Processo?

- IEEE
  - Sequência de passos executados com um determinado objetivo
- CMMI
  - Conjunto de ações e atividades inter-relacionadas realizadas para obter um conjunto especificado de produtos, resultados ou serviços.
- Processos ajudam a manter um nível de consistência, estrutura e qualidade nos projetos ou serviços produzidos por pessoas diferentes

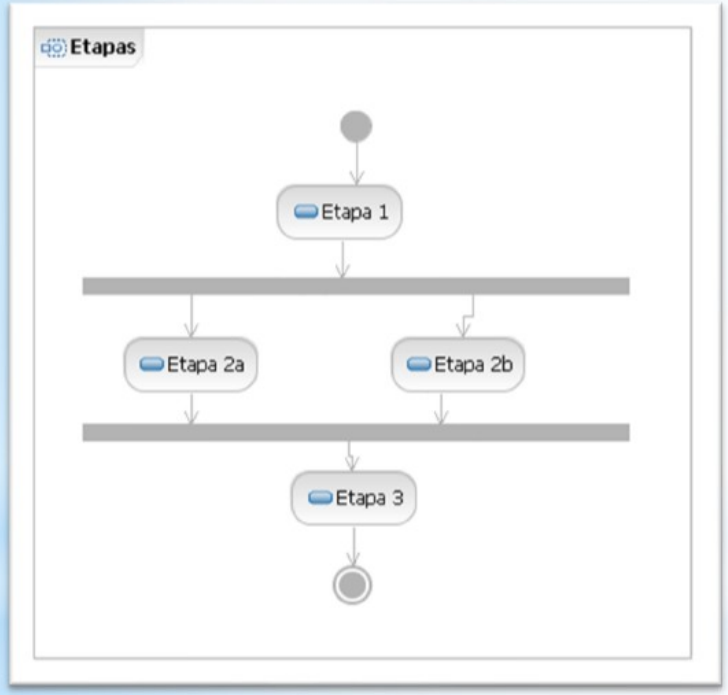
Estrutura de um Processo → Genérica

É um conjunto de passos parcialmente ordenados...

Porque permite paralelismo entre algumas etapas



Estrutura de um Processo → Genérica

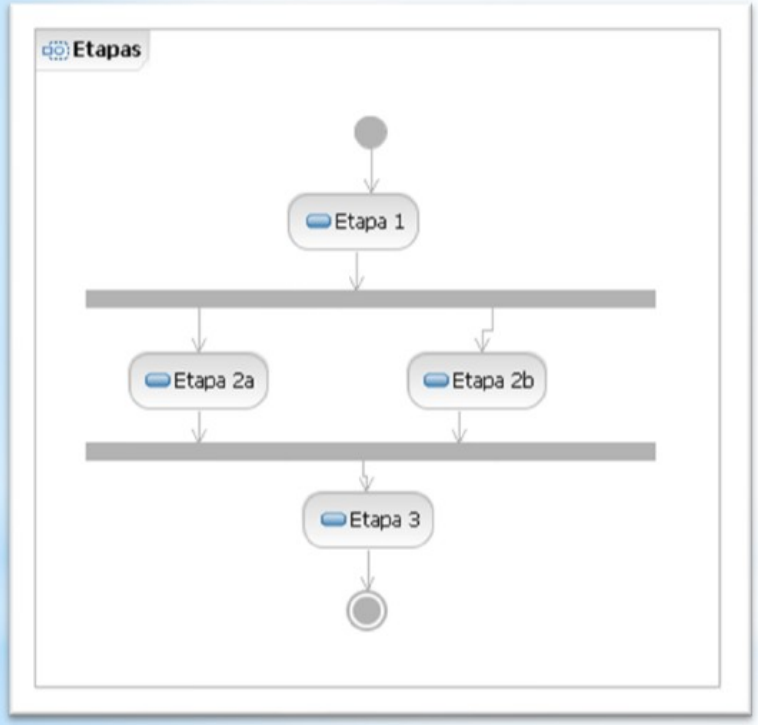


Entrega: É um artefato entregue ao cliente

Artefato é um produto "concreto" de trabalho do processo (e.g., modelo, documento ou código produzido por uma atividade)

- Artefato de Entrada - Atua como elemento de "entrada" para que se execute uma atividade
  - Ex: Lista de Requisitos e Necessidades do Cliente
- Artefato de Saída - Resultado produzido em uma atividade
  - Ex: Modelo do Problema(no domínio do Cliente)
- O artefato de saída de uma atividade, pode atuar como artefato de entrada para um etapa posterior.

Estrutura de um Processo → Genérica



Marco: Ponto de tempo que representa um estado significativo de um projeto, geralmente associado à conclusão e aprovação de um ou mais resultados

- Um processo deve estabelecer uma série de marcos
- Um marco é um ponto final de uma atividade de processo.
  - Ex: Aprovação da Especificação de Requisitos

Processo de Desenvolvimento de Software

- Um **processo de desenvolvimento de software** (***PDS***) compreende todas as **atividades necessárias** para definir, **desenvolver, testar e manter** um produto de **software** [Bezerra, 2007]

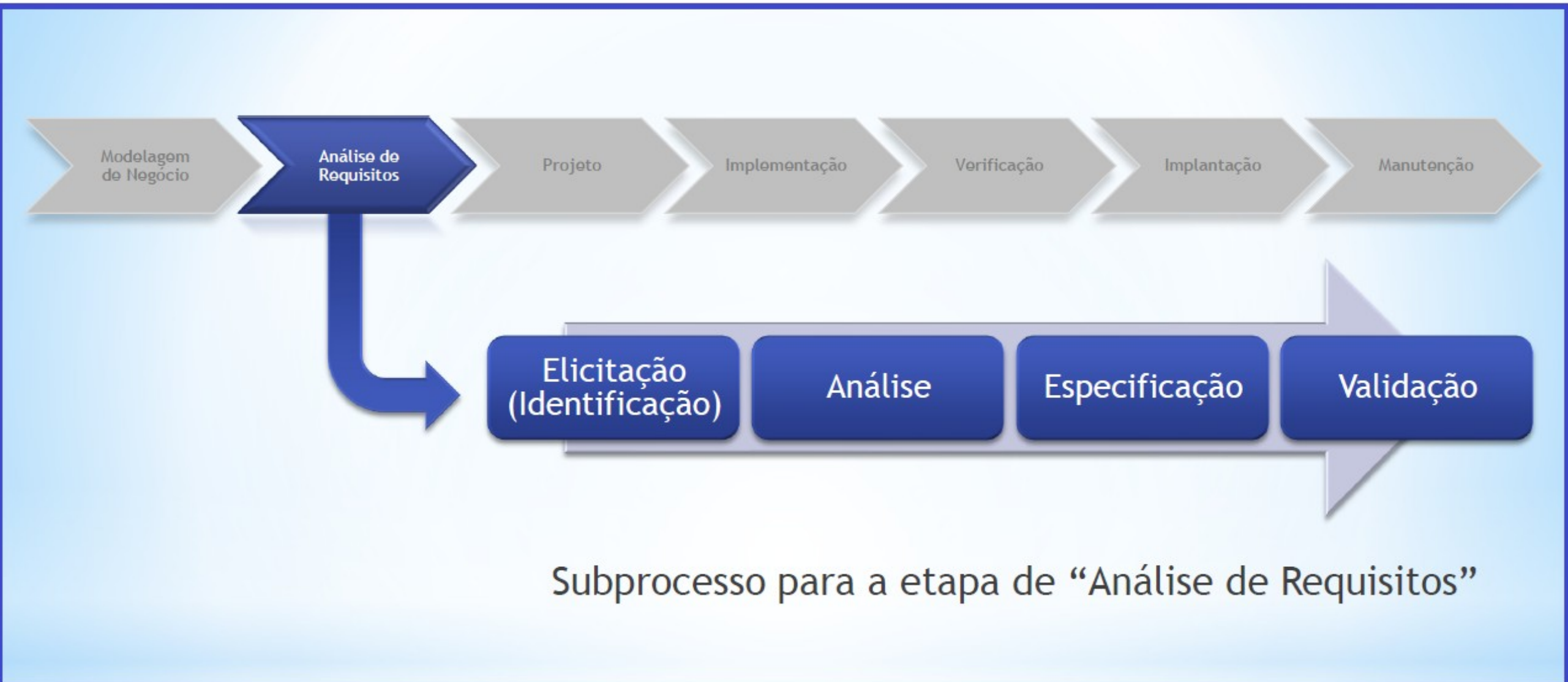
- Atividades genéricas em todos os processos:
  - ***Análise*** - o que o sistema deve fazer (funcionalidade) e quais as restrições
  - ***Desenvolvimento*** - produção do software
  - ***Verificação*** - avaliar correção, validação e outros aspectos de qualidade
  - ***Manutenção*** - mudanças no software

Processo de Desenvolvimento de Software

- Exemplo de um PDS

- Para cada etapa do processo exemplificado...
  - É possível executar um subprocesso...

Processo de Desenvolvimento de Software



Subprocesso para a etapa de “Análise de Requisitos”

Processo de Desenvolvimento de Software

Processo de Desenvolvimento de Software

- Processo definido tem documentação que detalha:
  - O que é feito (**Produto**)

Conjunto de artefatos entregues aos usuários finais e ao cliente em um projeto

Não está limitado somente ao software ou versão executável.

Envolve quaisquer artefatos intermediários necessários ao desenvolvimento e a comunicação do Engenheiro de Software com o cliente e os usuários

Processo de Desenvolvimento de Software

- Processo definido tem documentação que detalha:
  - O que é feito (Produto)
  - Quando (**Etapas**)

Divisão temporal de um processo

Processo de Desenvolvimento de Software

- Processo definido tem documentação que detalha:
  - O que é feito (Produto)
  - Quando (Etapas)
  - Por quem (**Papéis**)

Conjunto de proficiências, competências e responsabilidades, desempenhado por uma ou mais pessoas (e.g., gerente de projetos, analisa de requisitos, arquiteto de software...)

Cliente vs. **Usuário**

Processo de Desenvolvimento de Software

- Processo definido tem documentação que detalha:
  - O que é feito (Produto)
  - Quando (Etapas)
  - Por quem (Papéis)
  - As coisas que utilizam (**Insumos**)

Coisa consumida em um processo qualquer

Processo de Desenvolvimento de Software

- Processo definido tem documentação que detalha:
  - O que é feito (Produto)
  - Quando (Etapas)
  - Por quem (Papéis)
  - As coisas que utiliza (Insumos)
  - As coisas que produz em (**Resultados**)

Saída dos processos e atividades de um projeto, em contraposição aos Insumos

Processo de Desenvolvimento de Software

- Processo definido tem documentação que detalha:
  - O que é feito (Produto)
  - Quando (Etapas)
  - Por quem (Papéis)
  - As coisas que utiliza (Insumos)
  - As coisas que produz (Resultados)

- O ponto de partida para a arquitetura de um processo é a definição de um **Ciclo de Vida**

Ciclo de Vida de um Software

- Um ciclo de vida corresponde a um encadeamento específico das fases para construção de um sistema
  - Auxilia na forma de ordenação dos passos de um processo
- Alguns Modelos de Ciclo de Vida
  - Codifica-Remanda
  - Cascata
  - Cascata com Realimentação
  - Espiral
  - Prototipagem Evolutiva
  - Entrega Evolutiva

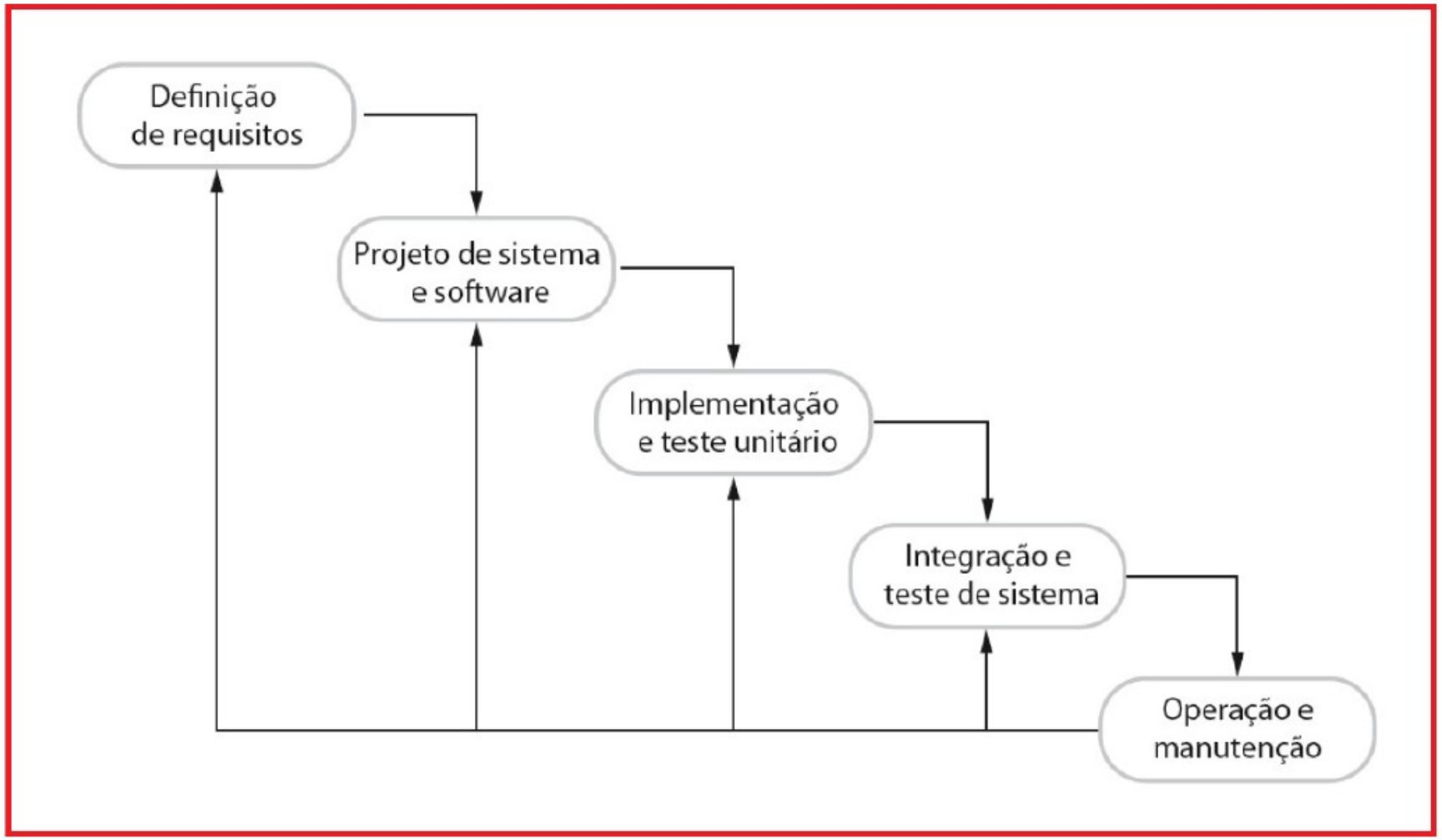
Modelo Codifica x Remenda

Modelo de ciclo de vida “Codifica-Remenda”

Modelo Codifica x Remenda

- Partindo apenas de uma especificação (ou nem isso), os desenvolvedores começam imediatamente a codificar,
  - Remendando à medida que os erros vão sendo descobertos
  - Provavelmente o mais usado
  - Não exige sofisticação técnica ou gerencial;
  - Alto risco;
  - Impossível de gerir;
  - Não permite assumir compromissos confiáveis.

Modelo Cascata



Modelo Cascata

Análise e definição de requisitos: Os serviços, restrições e metas do sistema são estabelecidos por meio de consulta aos usuários.

Projeto de sistema e software: Aloca os requisitos tanto para sistemas de hardware como para sistemas de software, por meio da definição de uma arquitetura geral do sistema.

Identificação e descrição das abstrações fundamentais do sistema de software e seus relacionamentos.

Modelo Cascata

Implementação e teste unitário: Desenvolvido como um conjunto de programas ou unidades de programa.

O teste unitário envolve a verificação de que cada unidade atenda a sua especificação.

Modelo Cascata

Integração e teste de sistema: As unidades individuais do programa são integradas e

Realiza-se testes como um sistema completo para assegurar que os requisitos do software tenham sido atendidos.

Entrega para o cliente.

Modelo Cascata

Operação e manutenção: O sistema é instalado e colocado em uso.

A manutenção envolve a correção de erros que não foram descobertos em estágios iniciais e possivelmente serão demonstrados novas necessidades.

Modelo Cascata

- Principais fases são executadas em estrita sequência
  - A fase seguinte só deve iniciar quando a anterior tiver sido concluída e aprovada pelas partes envolvida
  - Exemplo: O “Desenho” deve começar apenas quando a análise estiver totalmente concluída e aprovada
- Vantagens
  - Todo processo de desenvolvimento é estruturado
  - Permite demarcar pontos de controle que facilitam a gestão
  - Existem indícios de que essa abordagem é confiável e utilizável em projetos de qualquer escala

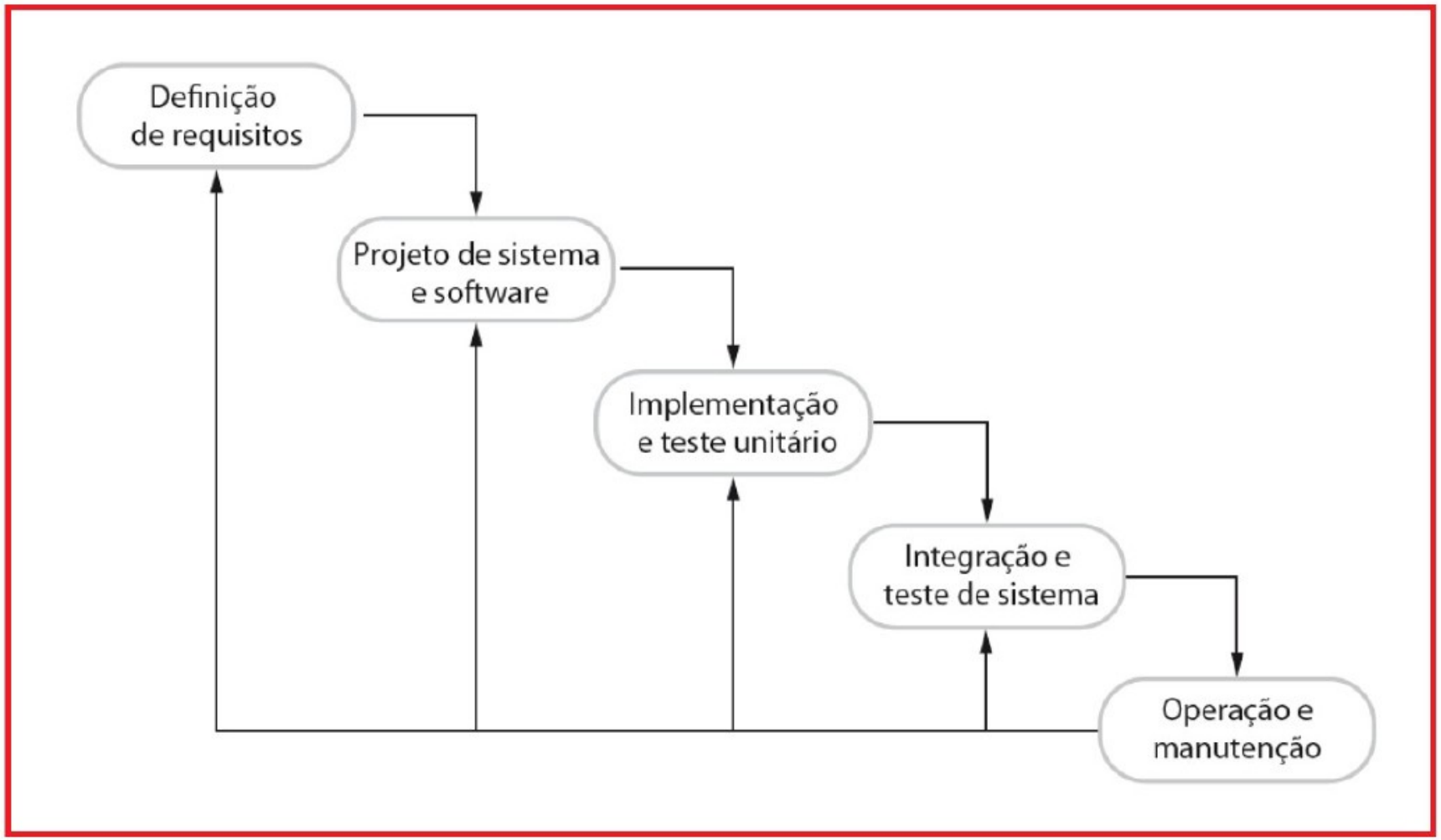
Modelo Cascata

- Desvantagens
  - Processo rígido e burocrático
  - Atividades de requisitos, análise e desenho têm que ser muito bem dominadas, não permitindo erros
    - Mas requisitos “sempre mudam” ☹
  - Dificuldade em realizar mudanças com o processo em andamento
    - Processo não prevê a correção posterior de problemas nas fases anteriores
  - Baixa visibilidade para o cliente
    - Só recebe o resultado final do projeto

Quando Aplicar o Modelo Cascata

- Sistemas críticos
- Quando os requisitos são bem compreendidos
- Quando há pouca probabilidade dos requisitos mudarem
- Projetos de engenharia de grandes sistemas onde o sistema é desenvolvido em vários locais.
  - Nessas circunstâncias, a natureza do modelo cascata dirigida a planos ajuda a coordenar o trabalho

Modelo Cascata com Realimentação



Cascata com Realimentação

- Permite a realimentação de correções entre fases
  - Permite que em **fases posteriores**, haja **revisão** e **alteração** de resultados das **fases anteriores.**
  - *Ex.: modelos e documentos de especificação e desenho podem ser alterados durante a implementação, à medida que problemas vão sendo descobertos*
- Desvantagem
  - A superposição das fases torna difícil gerenciar projetos baseados nesse modelo de ciclo de vida

Exercício de Fixação

- Julgue como Certo ou Errado:
- a. ( ) O modelo de ciclo de vida em cascata tem como características o estabelecimento, no início do projeto, de requisitos de maneira completa, correta e clara, e a possibilidade de disponibilização de várias versões operacionais do software antes da conclusão do projeto.

- b. ( ) O modelo de desenvolvimento em cascata é utilizado em caso de divergência nos requisitos de um software, para permitir a evolução gradual do entendimento dos requisitos durante a implementação do software.

- c. ( ) Durante a fase de levantamento de requisitos para a construção de um software, compete aos desenvolvedores organizar as necessidades em ordem de prioridade.

Modelos Evolutivos de Processos

- Existem situações em que a engenharia de software necessita de um modelo de processo que possa acomodar um produto que evolui com o tempo.

Modelos Evolutivos de Processos

- quando os requisitos de produto e de negócio mudam conforme o desenvolvimento procede
- quando uma data de entrega apertada (mercado) - impossível a conclusão de um produto completo
- quando um conjunto de requisitos importantes é bem conhecido, porém os detalhes ainda devem ser definidos

Modelos Evolutivos de Processos

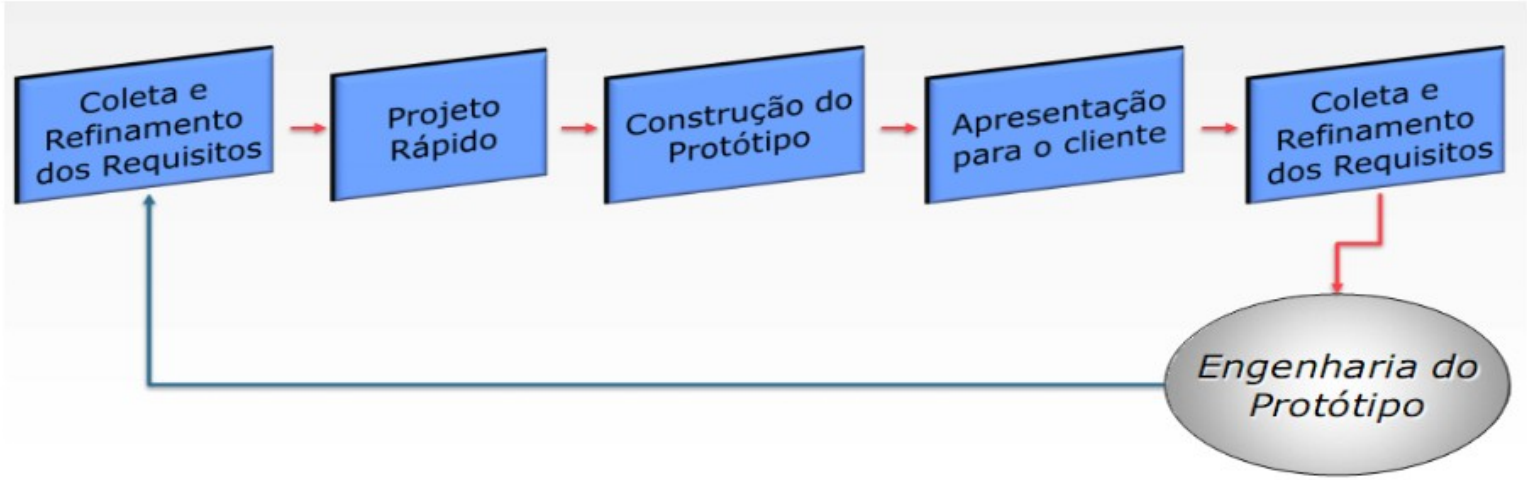
- modelos evolutivos são iterativos
- possibilitam o desenvolvimento de versões cada vez mais completas do software

Modelos Evolutivos de Processos

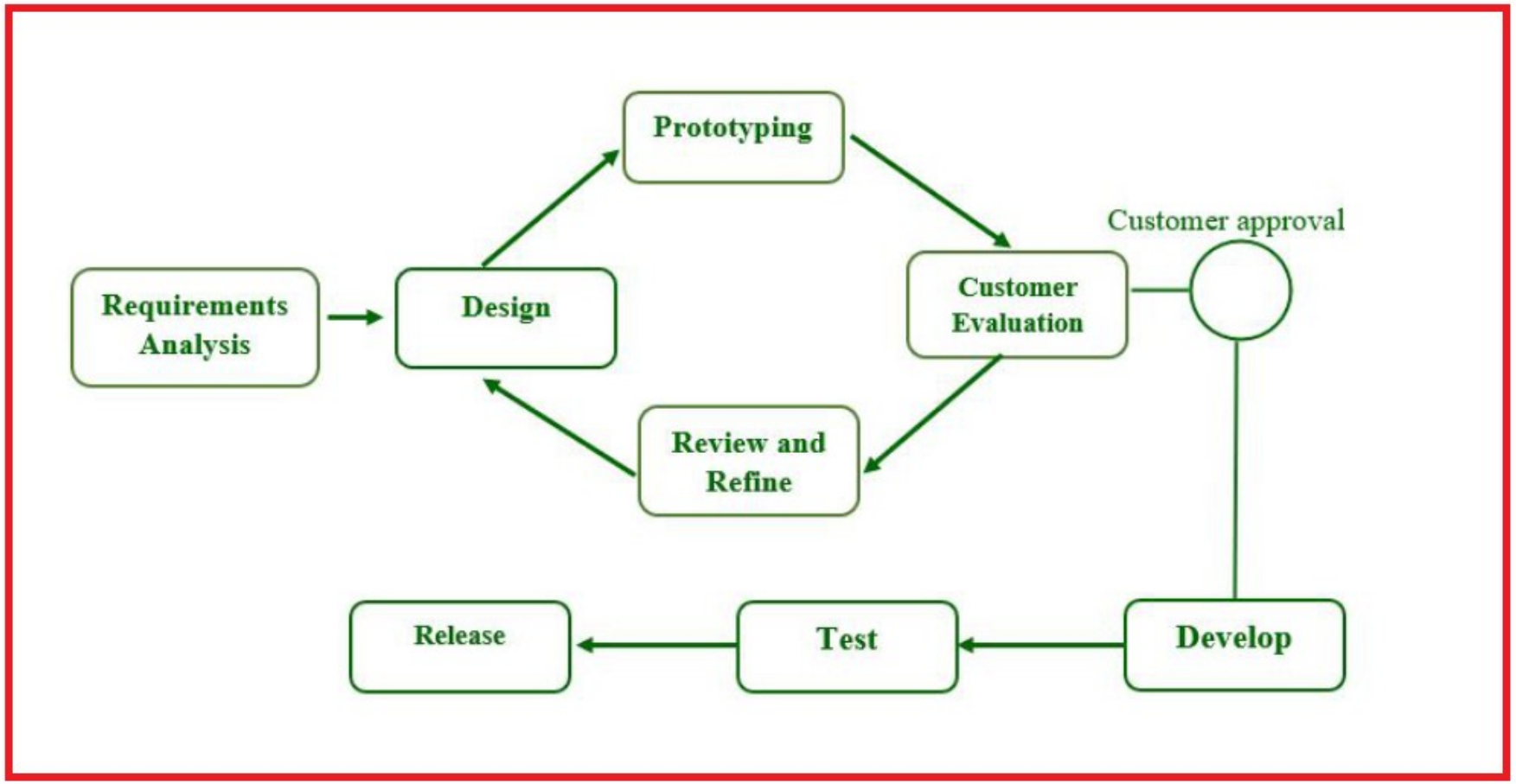
- Principais:
  - Prototipagem;
  - Incremental/Iterativo;
  - Espiral;
  - Modelo Orientado ao Reuso (componentes).

Modelo de Prototipagem

- Procura atender uma das limitações do modelo cascata: a impossibilidade de alterar requisitos.
- Consiste na criação de um modelo do software antes da implementação completa do sistema.
- Seleção de um conjunto de requisitos importantes para o cliente, ou que ainda não estão claros para os envolvidos no projeto.



Modelo de Prototipagem

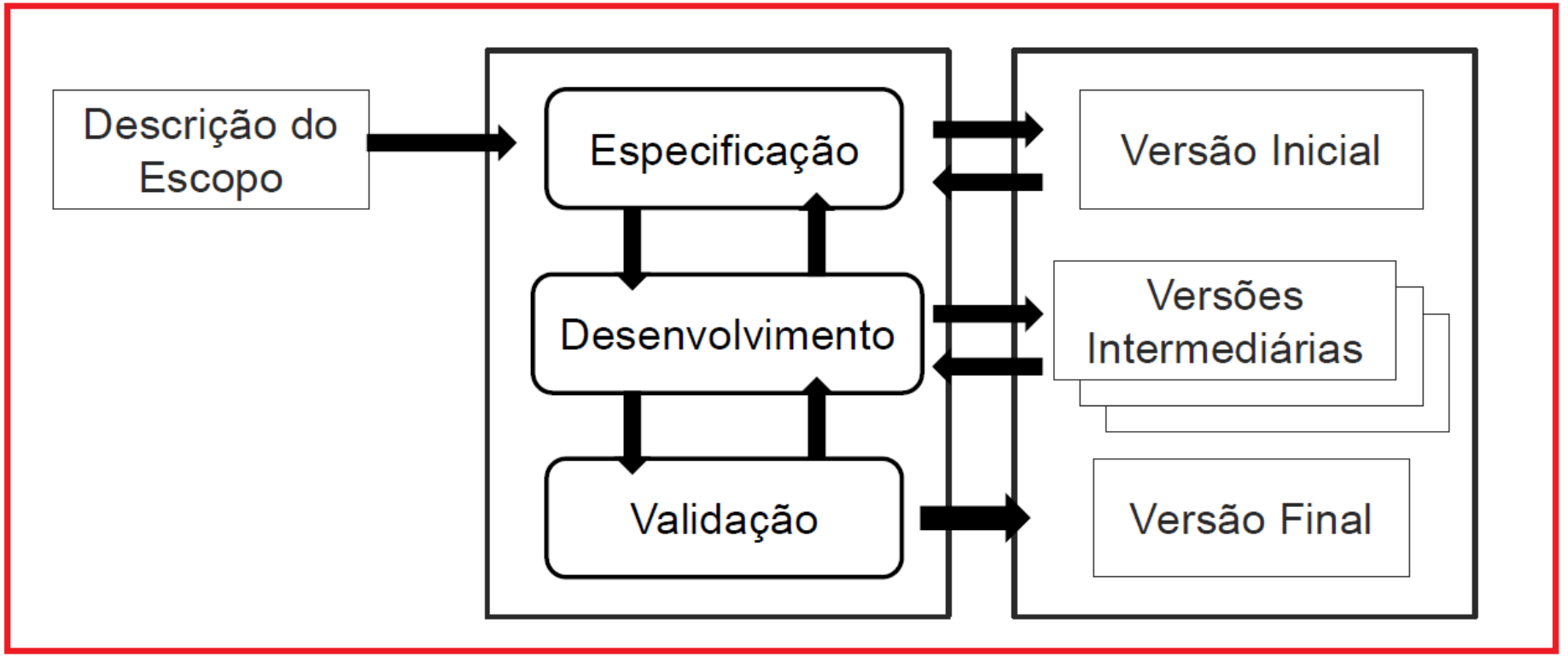


Modelo Incremental

- o modelo incremental combina elementos do modelo cascata (aplicado repetidamente) com a filosofia iterativa da prototipação
- o objetivo é trabalhar junto do usuário para descobrir seus requisitos, de maneira incremental, até que o produto final seja obtido.

Modelo Incremental

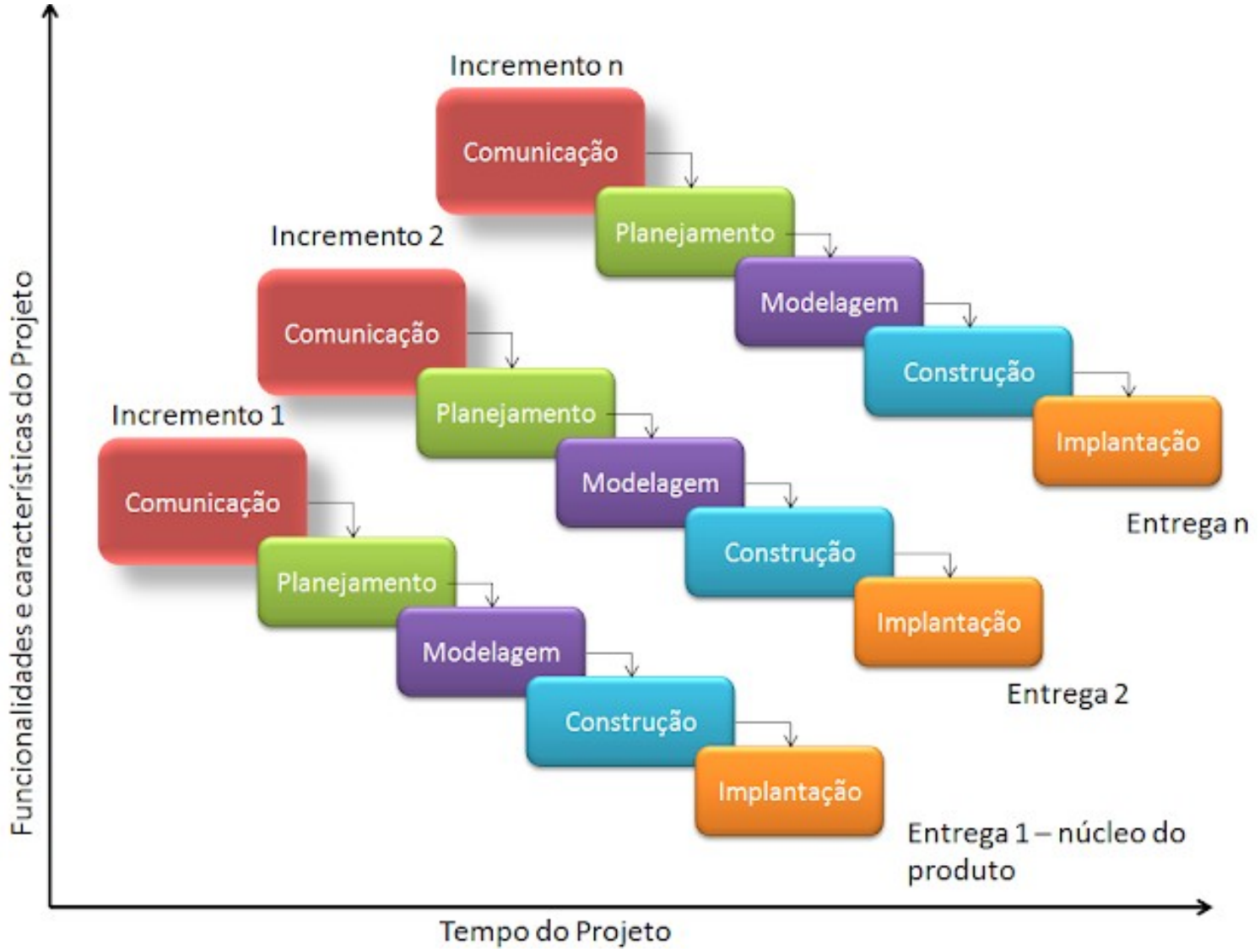
- Atividades são intercaladas
- Objetivo: dar feedback rápido ao cliente



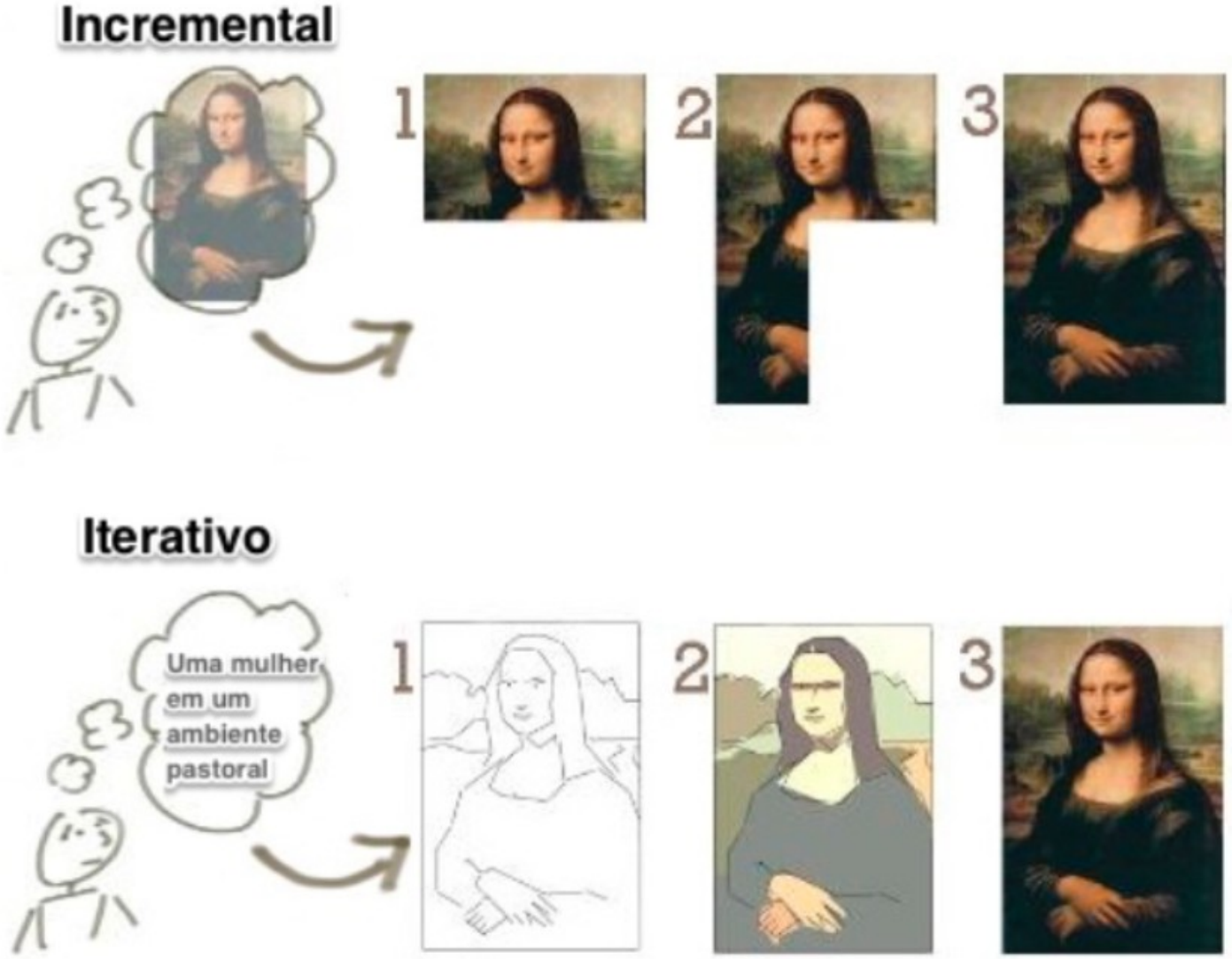
Modelo Incremental

- a versão inicial é frequentemente o núcleo do produto (a parte mais importante)
  - a evolução acontece quando novas características são adicionadas à medida que são sugeridas pelo usuário
- Este modelo é importante quando é difícil estabelecer *a priori* uma especificação detalhada dos requisitos

Modelo Incremental



Iterativo x Incremental



Iterativo x Incremental

- Uma diferença importante é se você especifica os requisitos de forma antecipada (incremental) ou os descobre ao construir seu produto (iterativo).
- Um processo de desenvolvimento de software é incremental quando a cada rodada é desenvolvido um pedaço inteiro do software. Já no iterativo, a cada iteração, se avança no conhecimento do projeto, novos requisitos são elicitados e a arquitetura do software é revisada.

Vantagens do Modelo Incremental

- Custo de acomodar mudanças nos requisitos é reduzido
- Mais fácil obter feedback do cliente
- Permite trabalhar com o cliente o entendimento dos requisitos
- Pode-se começar o sistema pelas partes melhor entendidas

Desvantagens do Modelo Incremental

- O processo pode não ser muito claro
- A gerência do software é complicada
  - O sistema não é completamente especificado à priori
- A estrutura do produto tende a se corromper com a adição de incrementos
  - O produto final é pode se tornar mal estruturado

Modelo Incremental

- o modelo incremental é mais apropriado para sistemas pequenos
- As novas versões podem ser planejadas de modo que os riscos técnicos possam ser administrados (Ex. disponibilidade de determinado hardware)

Exercícios de Fixação

- Assinale a alternativa que traz a informação correta sobre o modelo iterativo e o modelo incremental.

  a) O modelo iterativo sempre disponibiliza uma versão que pode ser utilizada pelo usuário final.

  b) No modelo incremental, iterações são realizadas e, ao fim de cada uma delas, diversas partes do sistema podem ter sido desenvolvidas e não concluídas.

  c) O modelo cascata e o modelo incremental são muito semelhantes, uma vez que, em ambos ciclos de desenvolvimento, um módulo só pode ser entregue após a conclusão dos demais que já estavam em andamento.

  d) O modelo iterativo deixa livre que diversos módulos sejam iniciados na mesma iteração sem a devida conclusão ao fim do ciclo.

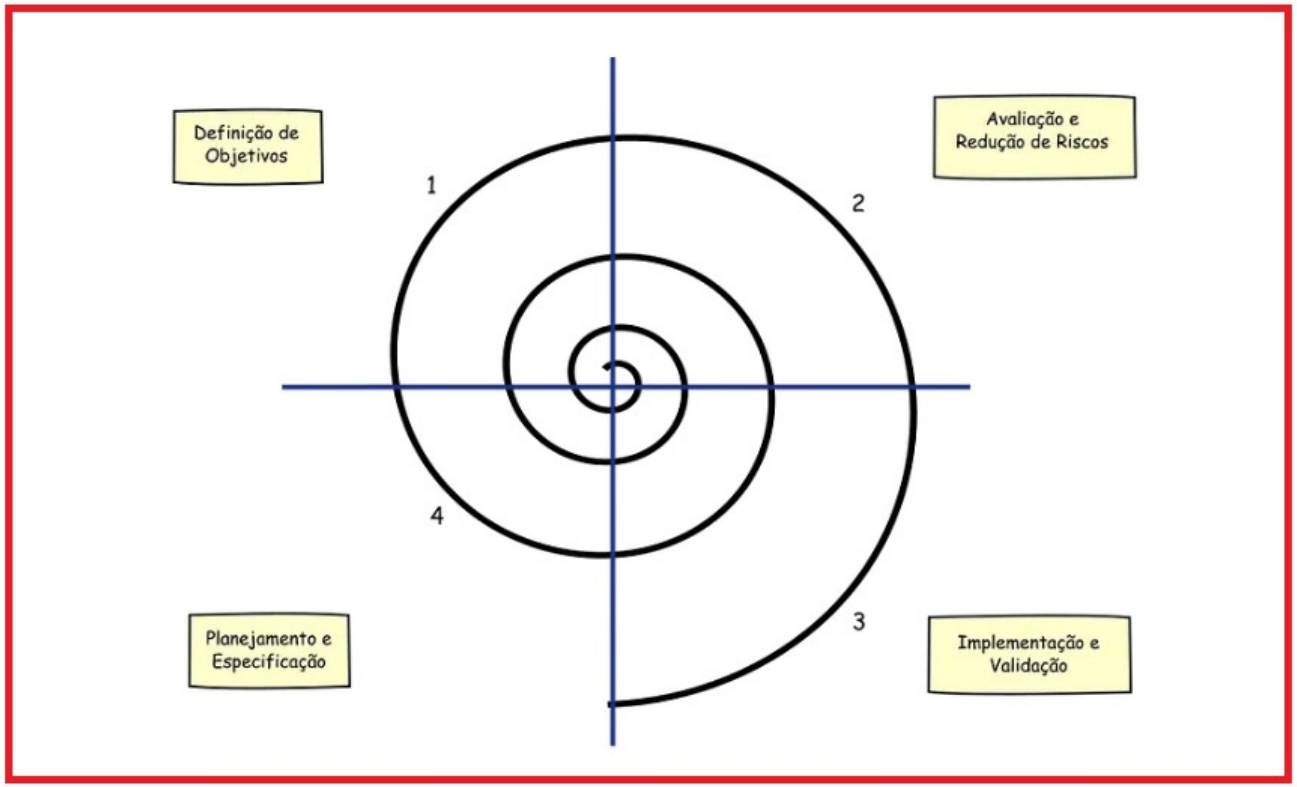
Modelo Espiral

- Cada ciclo do espiral é uma fase do processo
- Cada ciclo determina quatro etapas fundamentais:
  - Análise e redução de riscos
  - Desenvolvimento e validação
  - Planejamento do próximo ciclo
  - Definição de objetivos, alternativas e restrições
- Protótipos são construídos em cada ciclo.
- Não há fases fixas pré-definidas. Elas são definidas de acordo com os objetivos

Modelo Espiral

- Fases do Processo

Cada “loop” do espiral é dividido em 4 setores

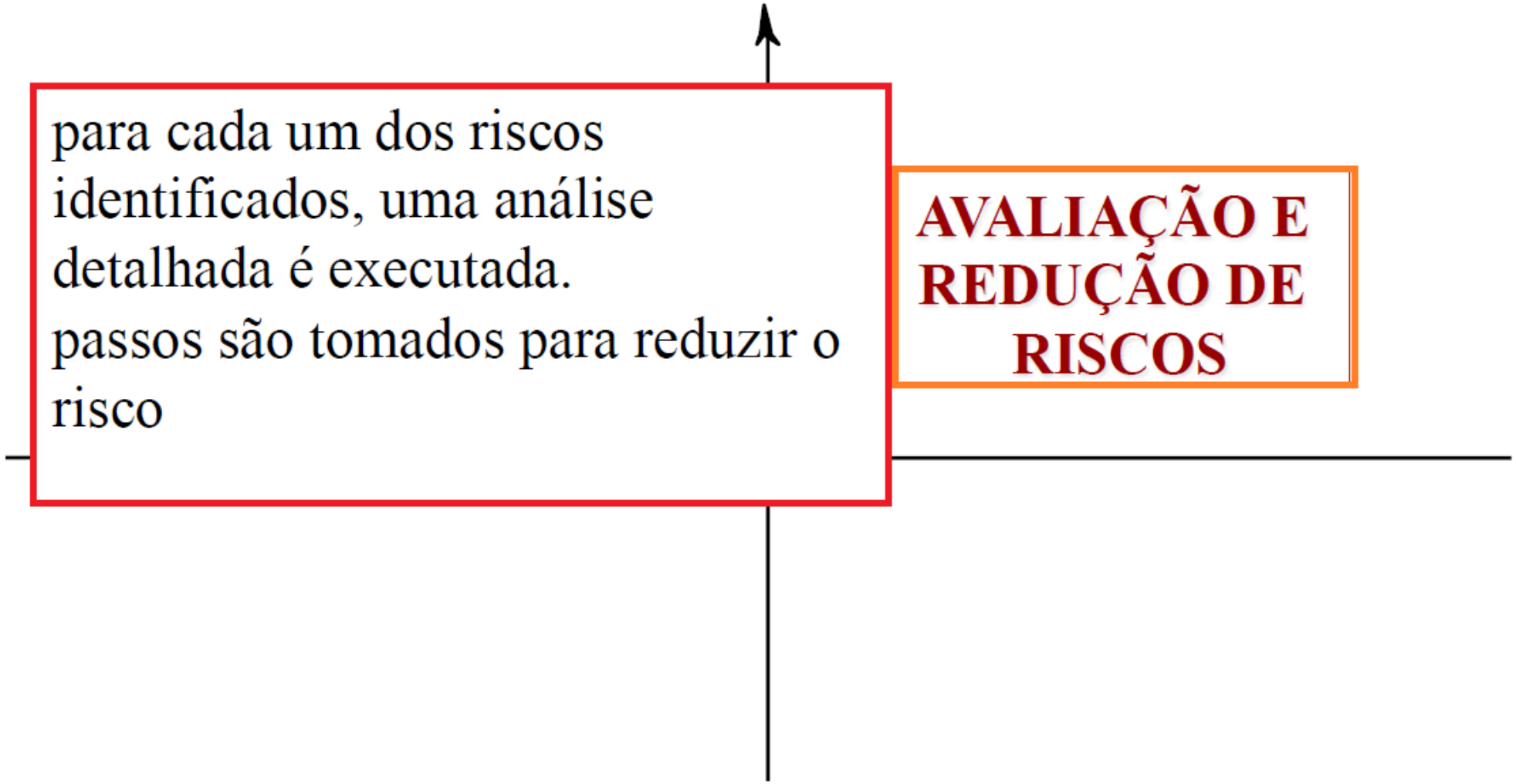


Modelo Espiral (com 4 regiões)

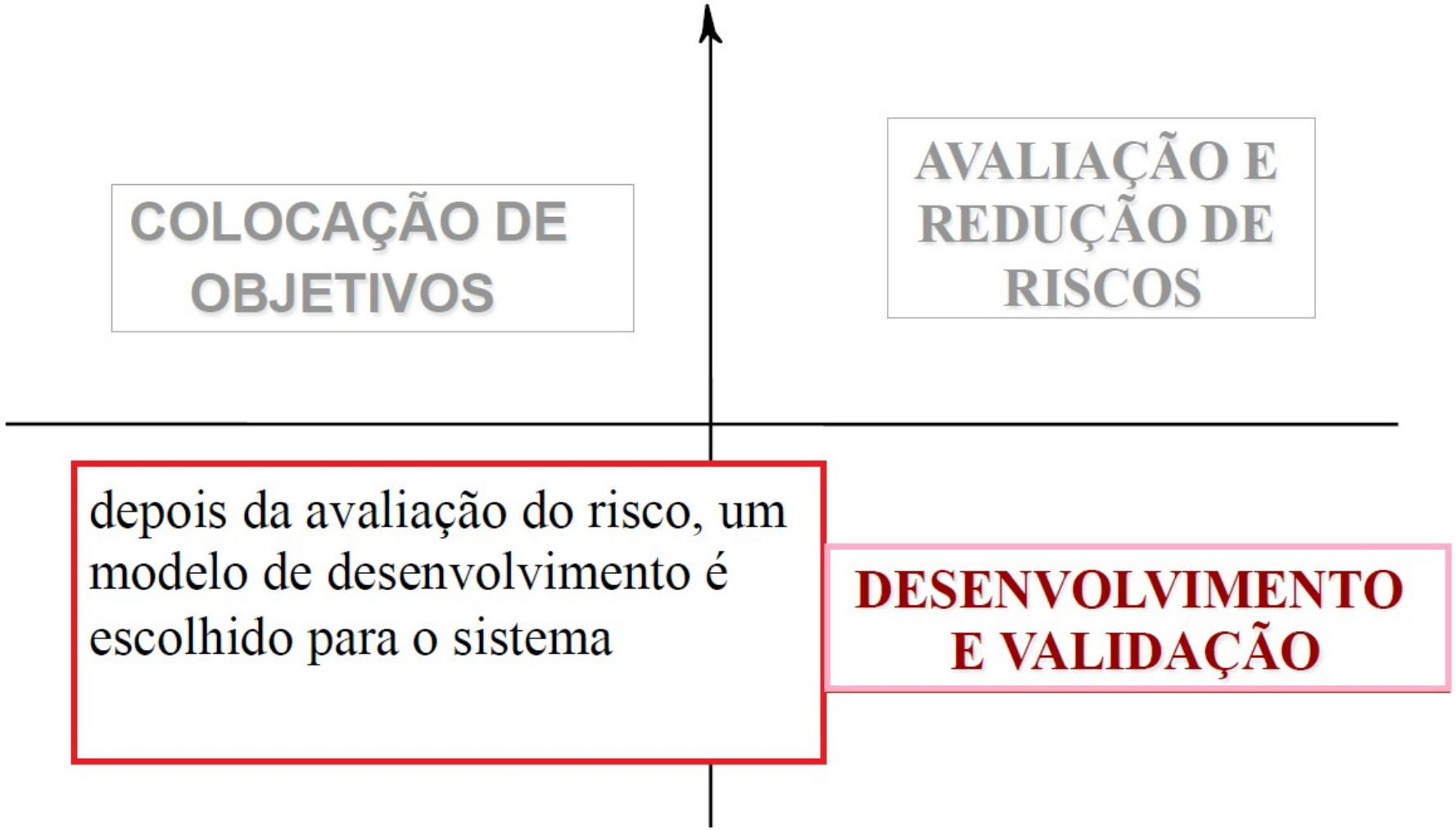
COLOCAÇÃO DE OBJETIVOS

são definidos objetivos específicos para a fase do projeto
são identificadas restrições sobre o processo e o produto
é projetado um plano de gerenciamento detalhado
são identificados riscos do projeto
dependendo dos riscos, estratégias alternativas podem ser planejadas

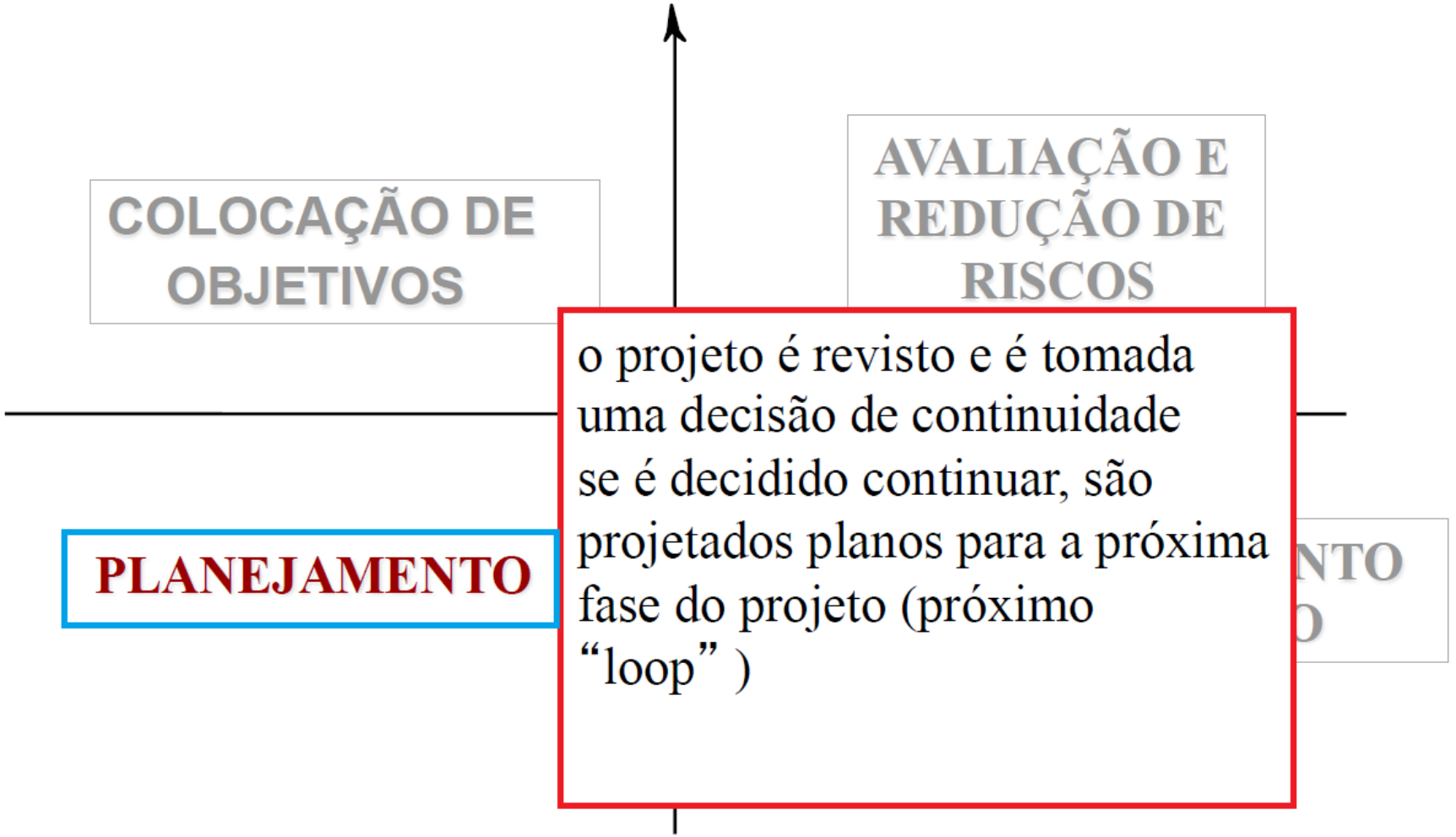
Modelo Espiral (com 4 regiões)



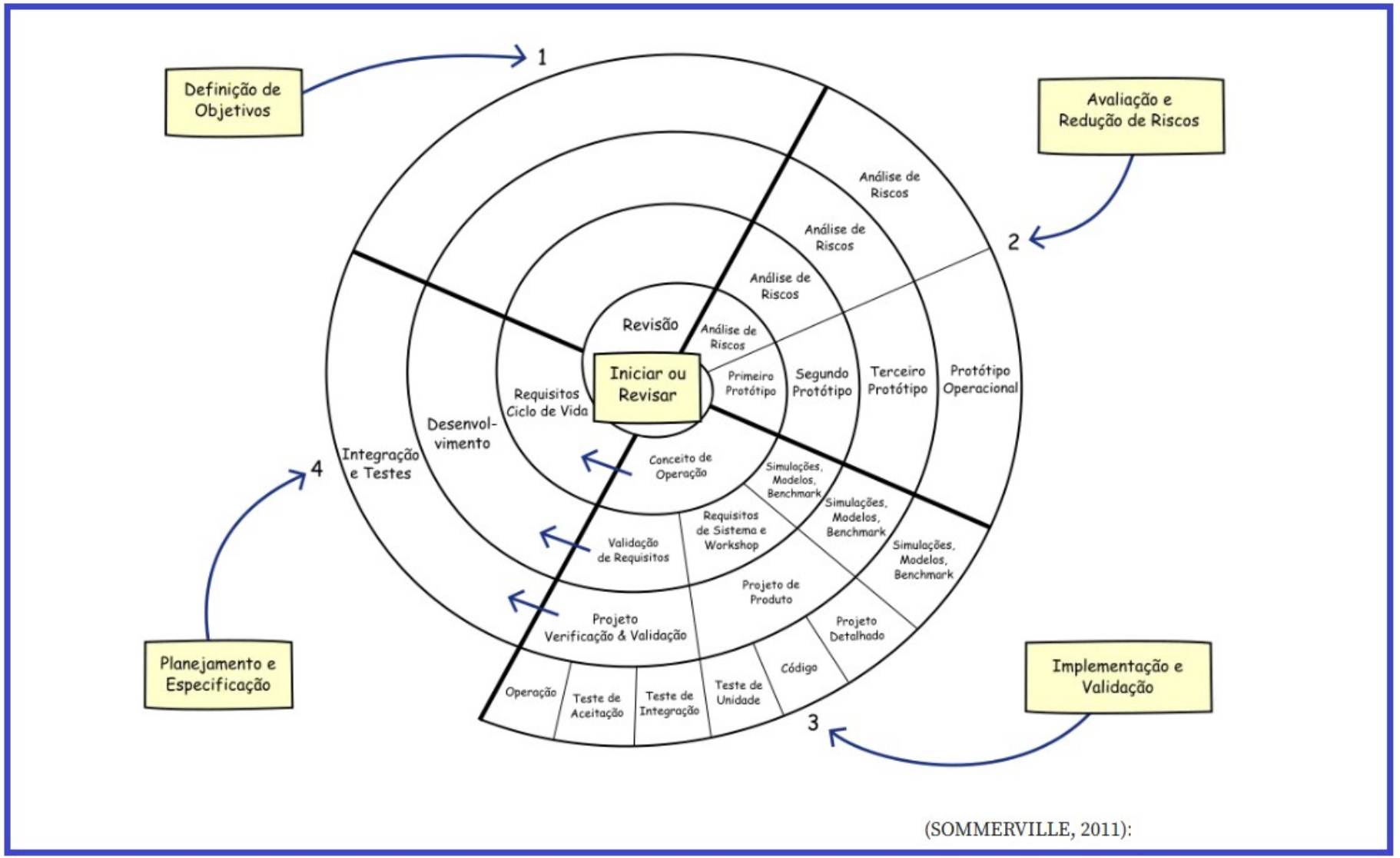
Modelo Espiral (com 4 regiões)



Modelo Espiral (com 4 regiões)

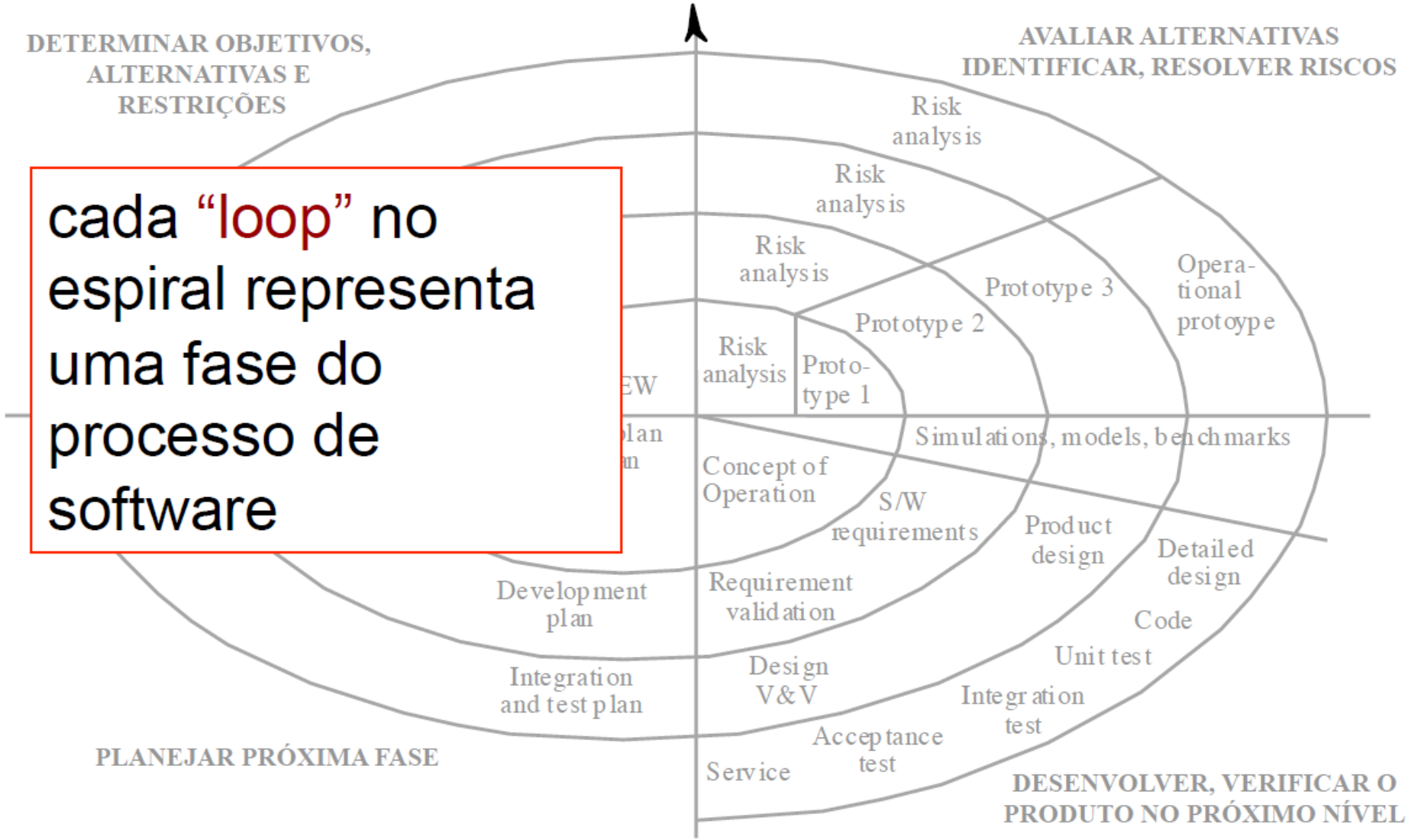


Modelo Espiral (com 4 regiões)



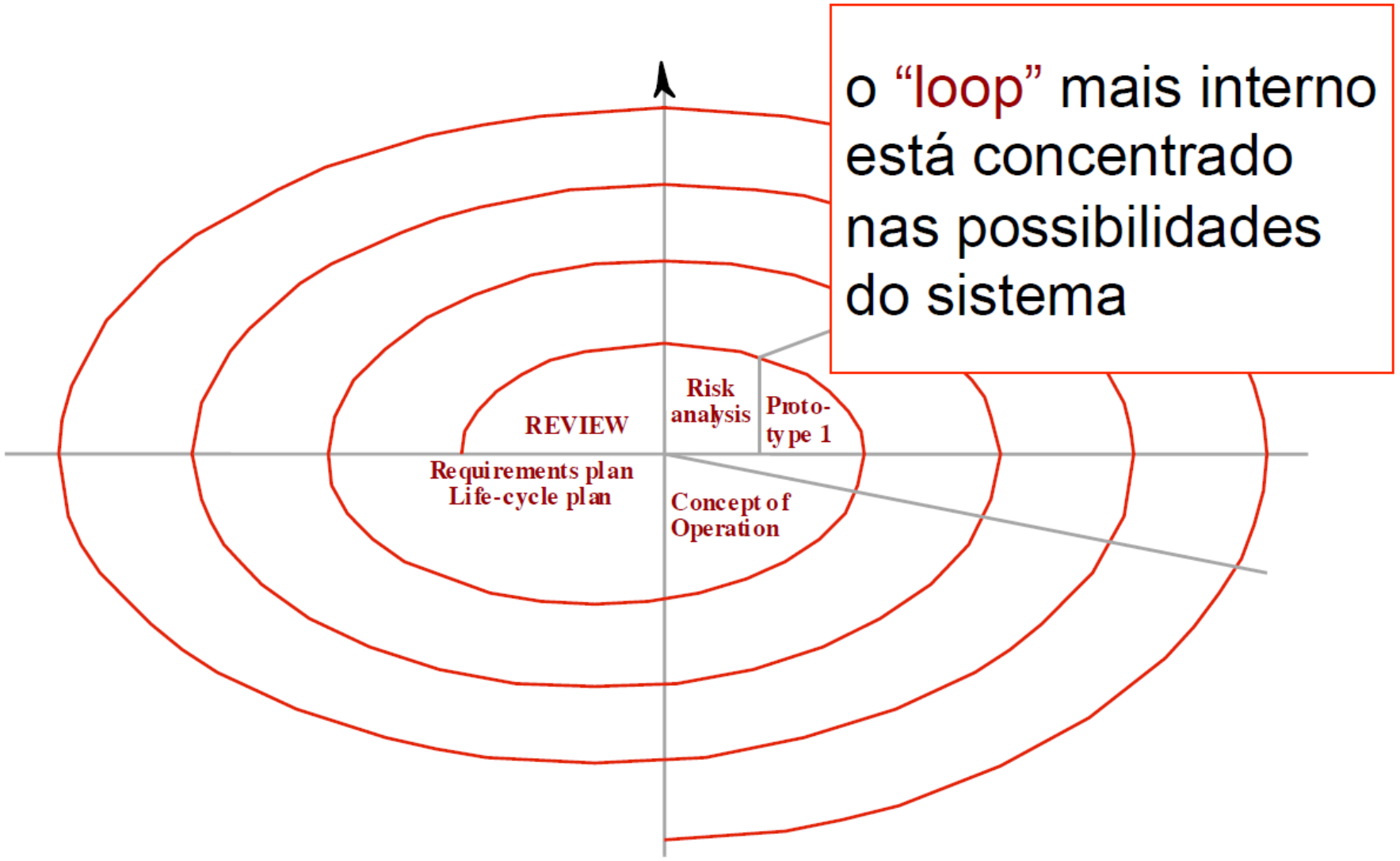
(SOMMERVILLE, 2011):

Modelo Espiral (com 4 regiões)



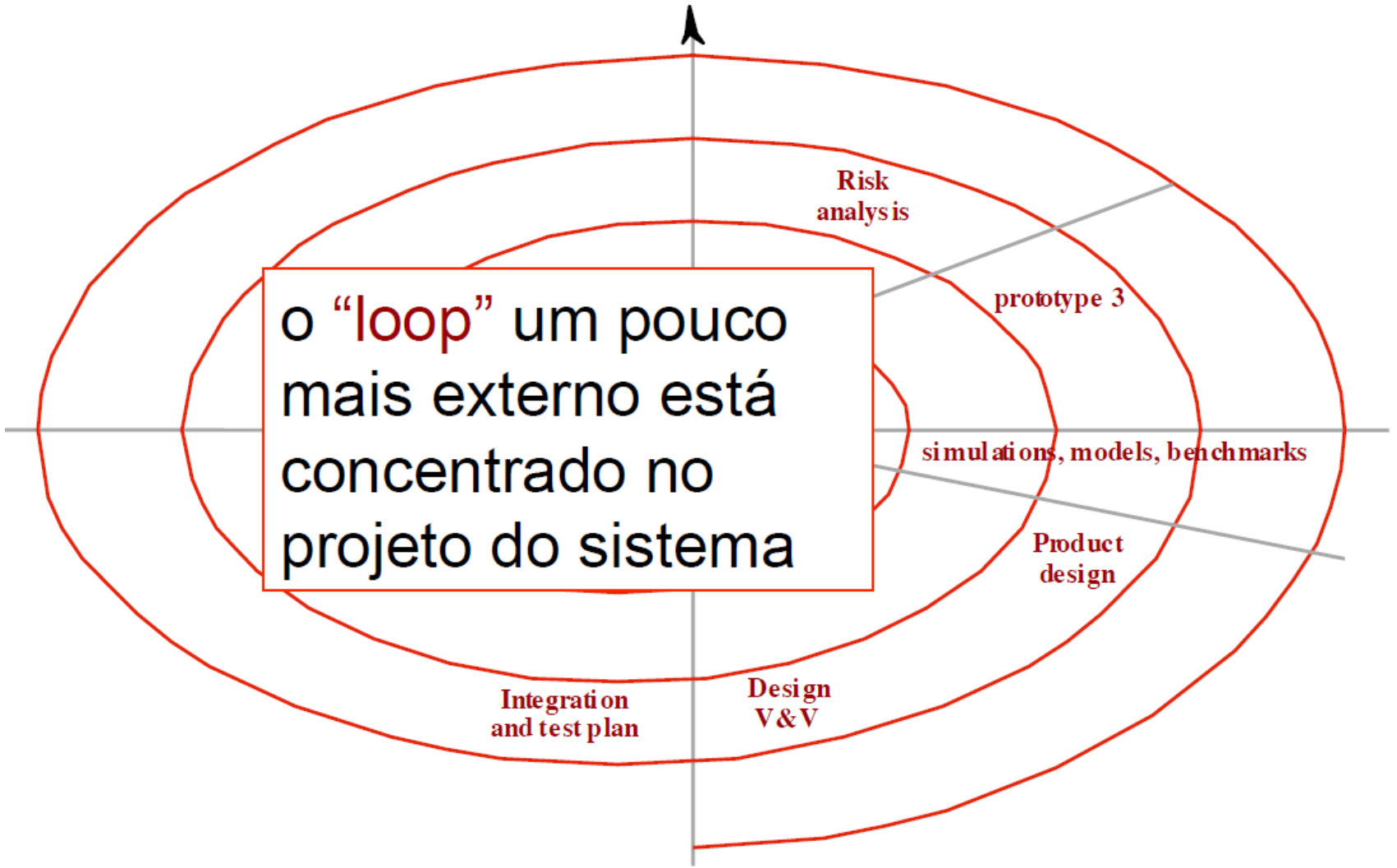
cada "loop" no espiral representa uma fase do processo de software

Modelo Espiral (com 4 regiões)



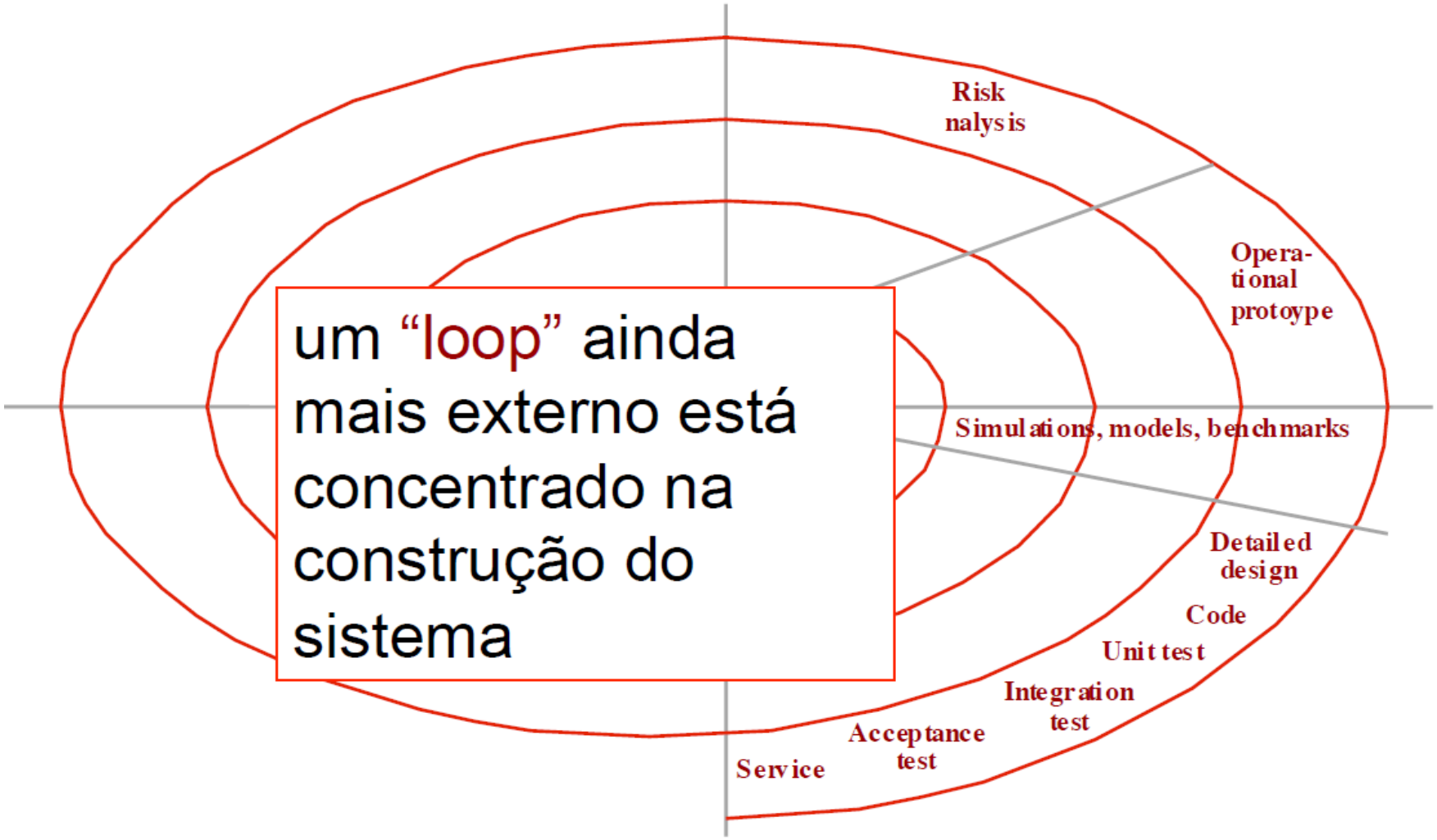
Modelo Espiral (com 4 regiões)

Modelo Espiral (com 4 regiões)



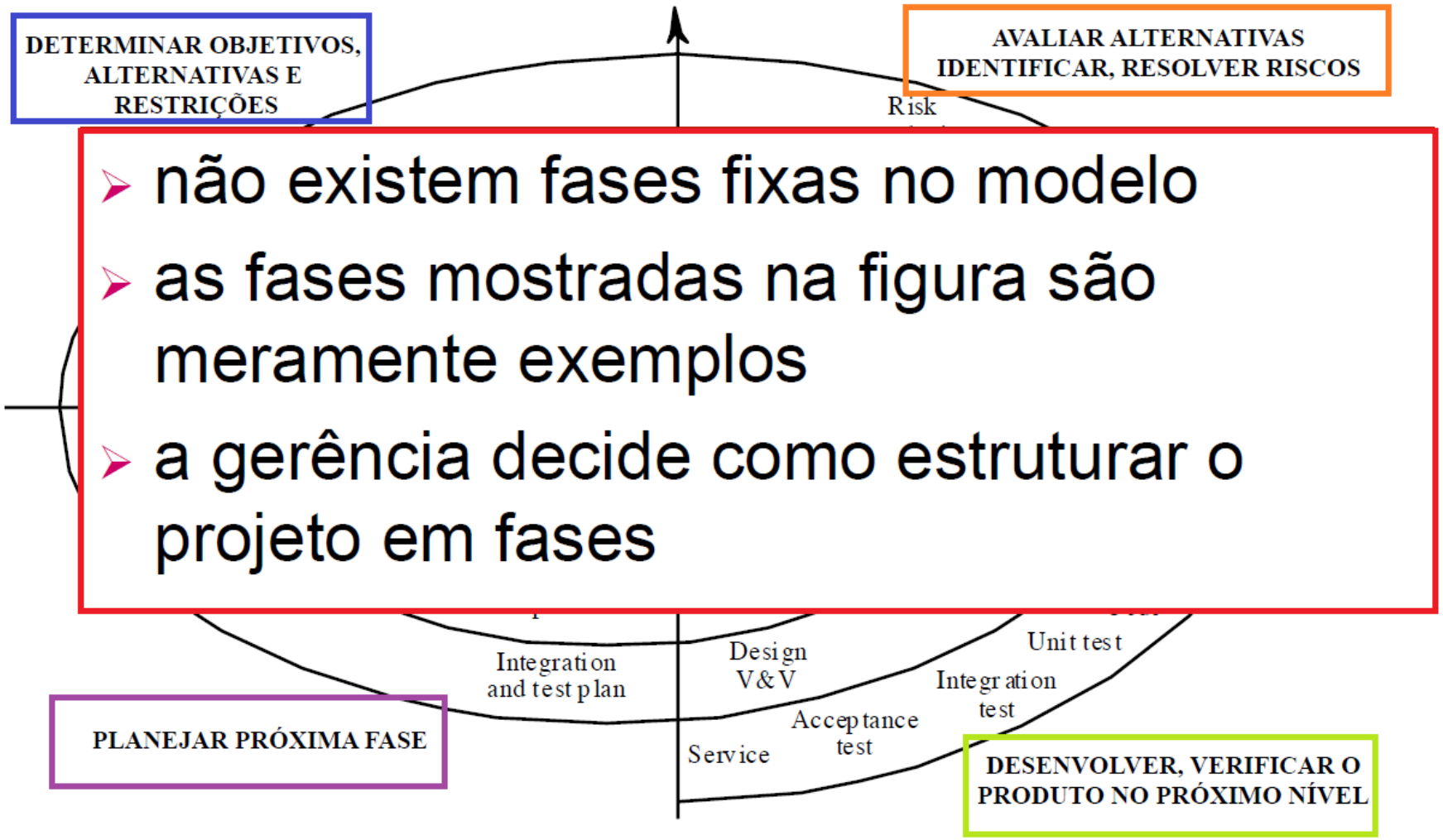
o "loop" um pouco mais externo está concentrado no projeto do sistema

Modelo Espiral (com 4 regiões)



um “loop” ainda mais externo está concentrado na construção do sistema

Modelo Espiral (com 4 regiões)



- não existem fases fixas no modelo
- as fases mostradas na figura são meramente exemplos
- a gerência decide como estruturar o projeto em fases

Comentários sobre o Modelo Espiral

- é, atualmente, a abordagem mais realística para o desenvolvimento de software em grande escala.
- usa uma abordagem que capacita o desenvolvedor e o cliente a entender e reagir aos riscos em cada etapa evolutiva
- pode ser difícil convencer os clientes que uma abordagem "evolutiva" é controlável

Comentários sobre o Modelo Espiral

- exige considerável experiência na determinação de riscos e depende dessa experiência para ter sucesso
- o modelo é relativamente novo e não tem sido amplamente usado. Demorará muitos anos até que a eficácia desse modelo possa ser determinada com certeza absoluta

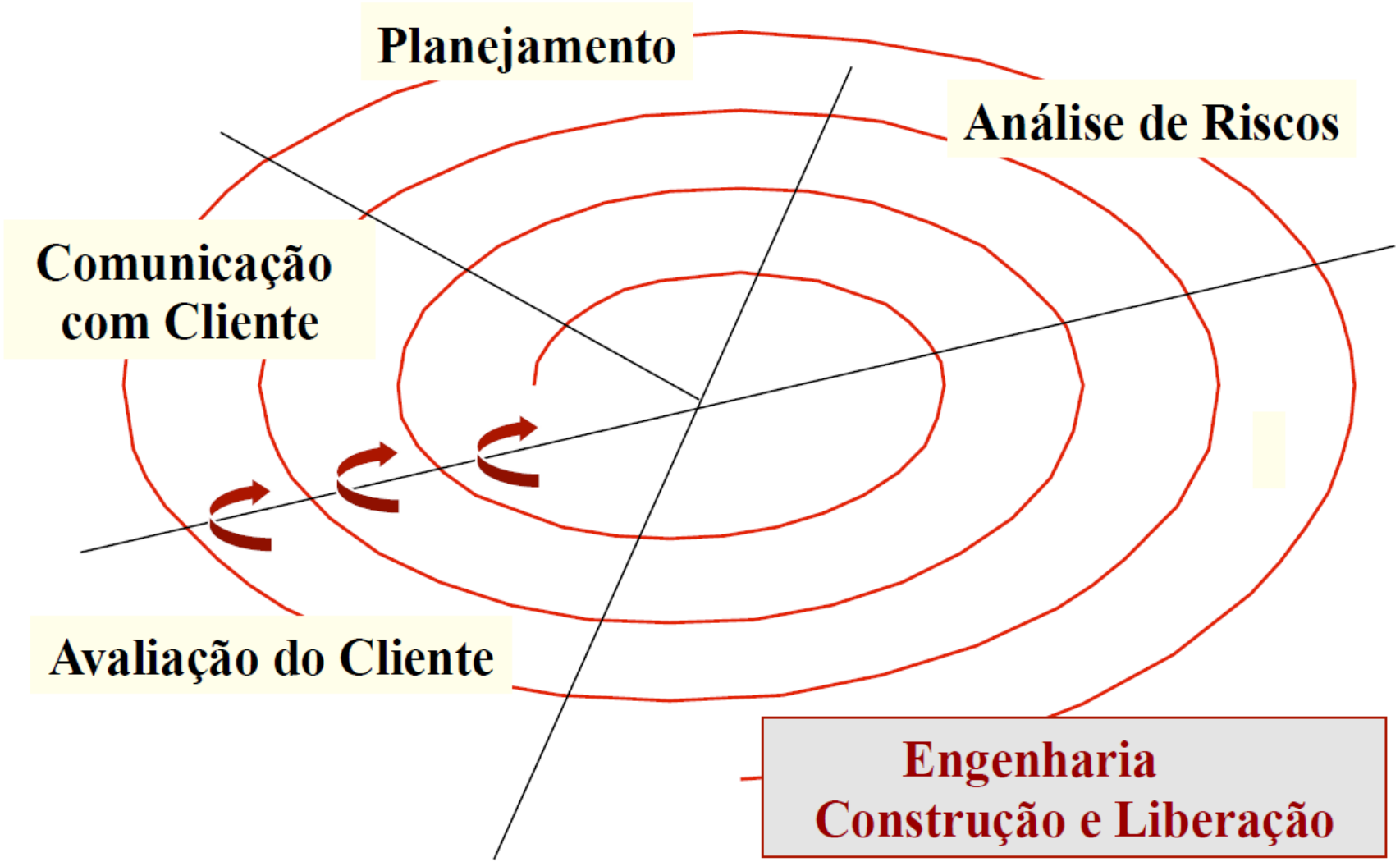
Modelo Orientado a Reuso

- Caracteriza-se pela criação de uma aplicação a partir de componentes previamente construídos ou COTS (*Commercial-of-the-shelf).*
- Baseia-se em uma grande base de componentes e um framework para integração deles.
- Conceito de *Framework:*
  - “ Um framework ou arcabouço é uma estrutura de suporte definida em que um outro projeto de software pode ser organizado e desenvolvido. Um framework pode incluir programas de suporte, bibliotecas de código, linguagens de script e outros softwares para ajudar a desenvolver e juntar diferentes componentes de um projeto de software”.

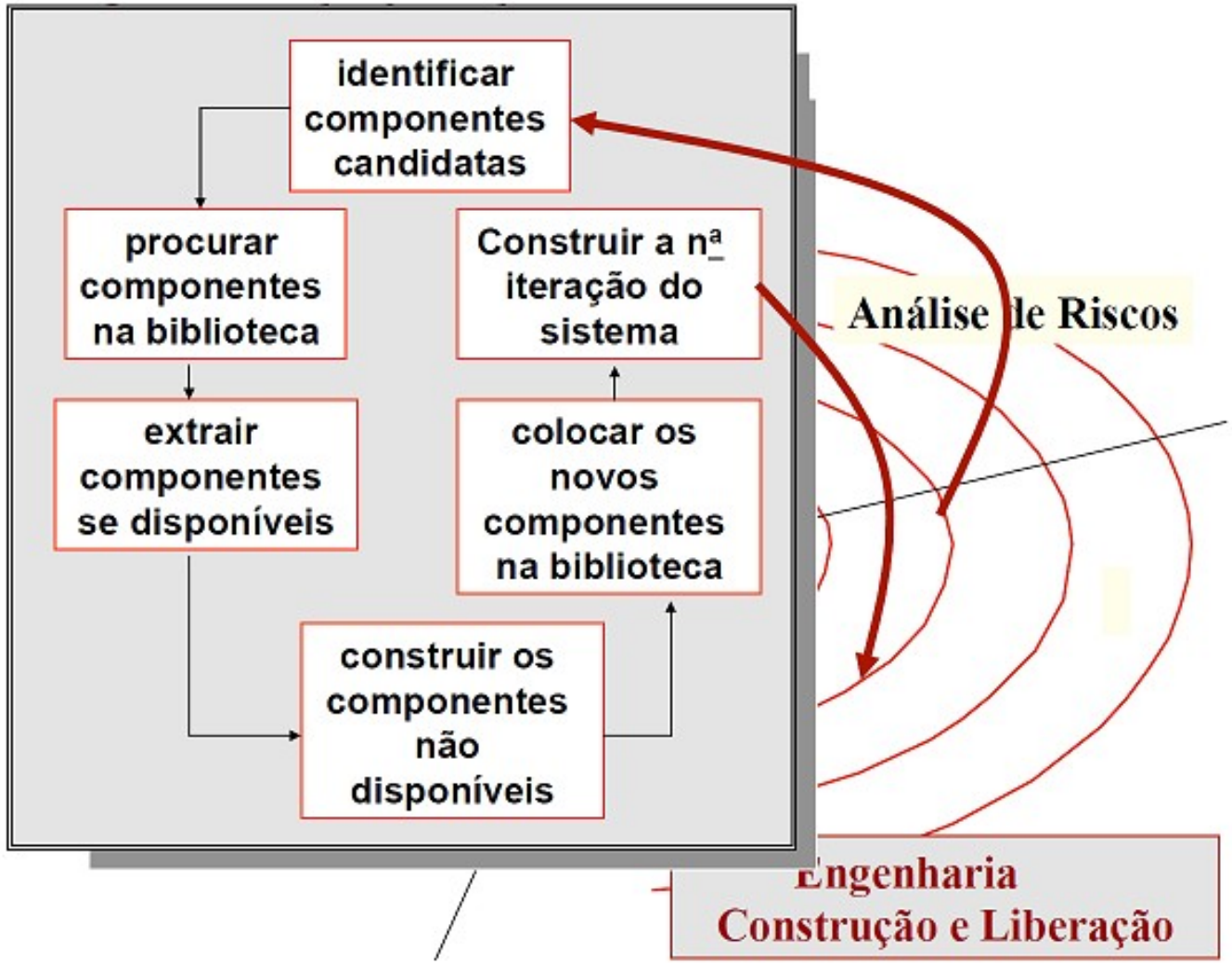
Modelo Orientado ao Reuso

- Utiliza tecnologias orientadas a objeto
- Quando projetadas e implementadas apropriadamente as classes orientadas a objeto são reutilizáveis em diferentes aplicações e arquiteturas de sistema
- O modelo de montagem de componentes incorpora muitas das características do modelo espiral.

Modelo Orientado ao Reuso



Modelo Orientado ao Reuso



Modelo Orientado ao Reuso

- O modelo de montagem de componentes conduz ao reuso do software
- a reusabilidade fornece uma série de benefícios:
  - redução de 70% no tempo de desenvolvimento
  - redução de 84% no custo do projeto
  - índice de produtividade de 26.2 (normal da indústria é de 16.9)
- esses resultados dependem da robustez da biblioteca de componentes

Modelo RAD

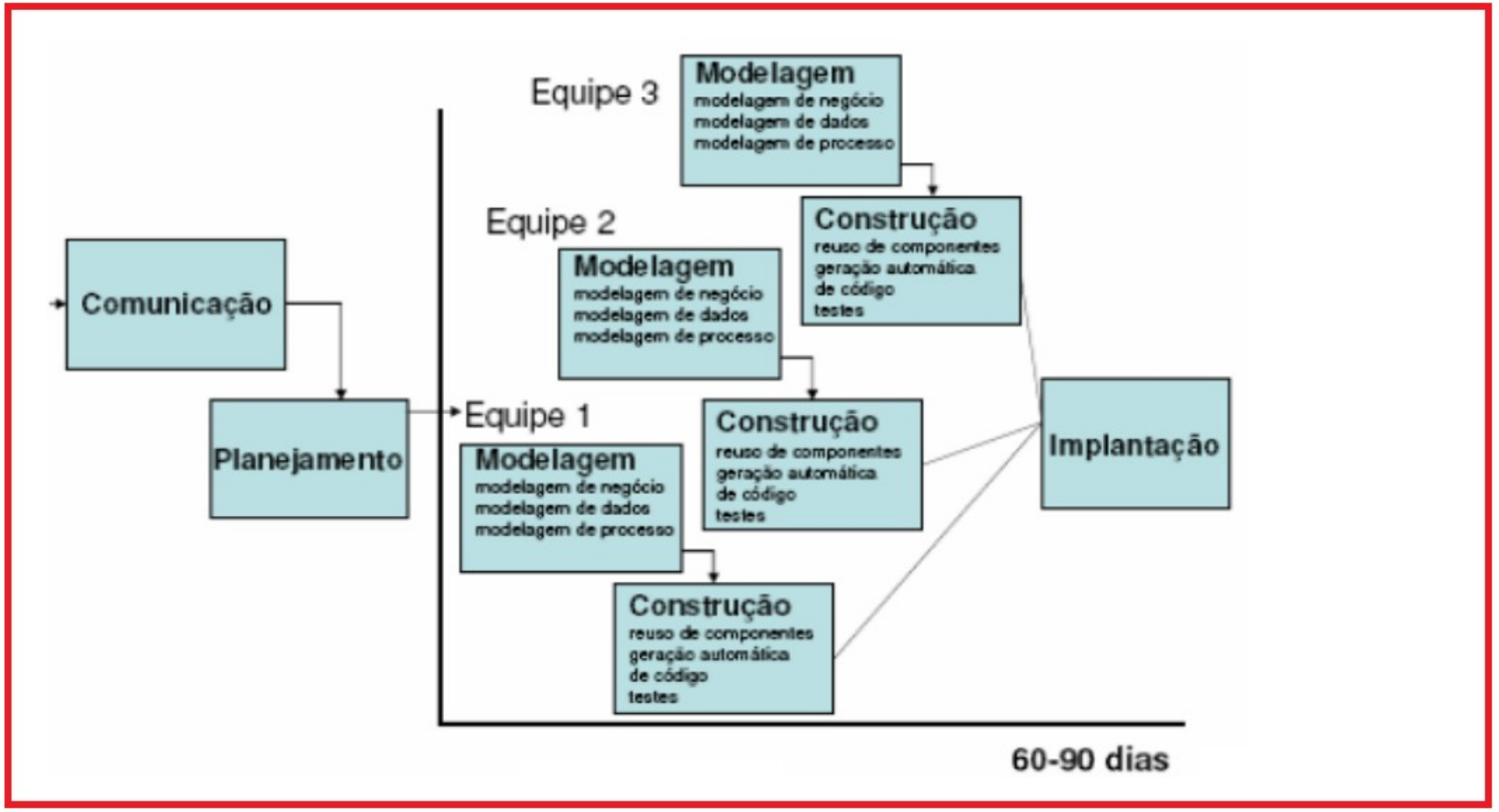
RAD (Rapid Application Development)

- Abordagem baseada em componentes.
- Usado principalmente para aplicações de SI.
- Há equipes separadas que no final se integram.
- Sequencial linear (enfatizando um ciclo)

DESVANTAGENS:

- Necessário número de desenvolvedores suficientes para as equipes.
- Comprometimento entre cliente e desenvolvedor para prazos curtos.

Modelo RAD


Equipe 3
Modelagem
modelagem de negócio
modelagem de dados
modelagem de processo
Construção
reuso de componentes
geração automática
de código
testes
Equipe 2
Modelagem
modelagem de negócio
modelagem de dados
modelagem de processo
Comunicação
Planejamento
Equipe 1
Modelagem
modelagem de negócio
modelagem de dados
modelagem de processo
Construção
reuso de componentes
geração automática
de código
testes
Implantação
Construção
reuso de componentes
geração automática
de código
testes
60-90 dias

Exercício 5

- Justificando sua resposta com base no tipo de sistema a ser desenvolvido, sugira o modelo de processo de software mais apropriado que poderia ser usado como base para gerenciar o desenvolvimento dos seguintes sistemas:
  - a. um sistema para controlar um antibloqueador de freios de um automóvel;
  - b. um sistema de realidade virtual para apoiar a manutenção de carros;
  - c. Um sistema de contabilidade de uma universidade, que substituirá um já existente;
  - d. Um sistema interativo que permite aos passageiros encontrar os horários dos trens por meio de terminais instalados nas estações.

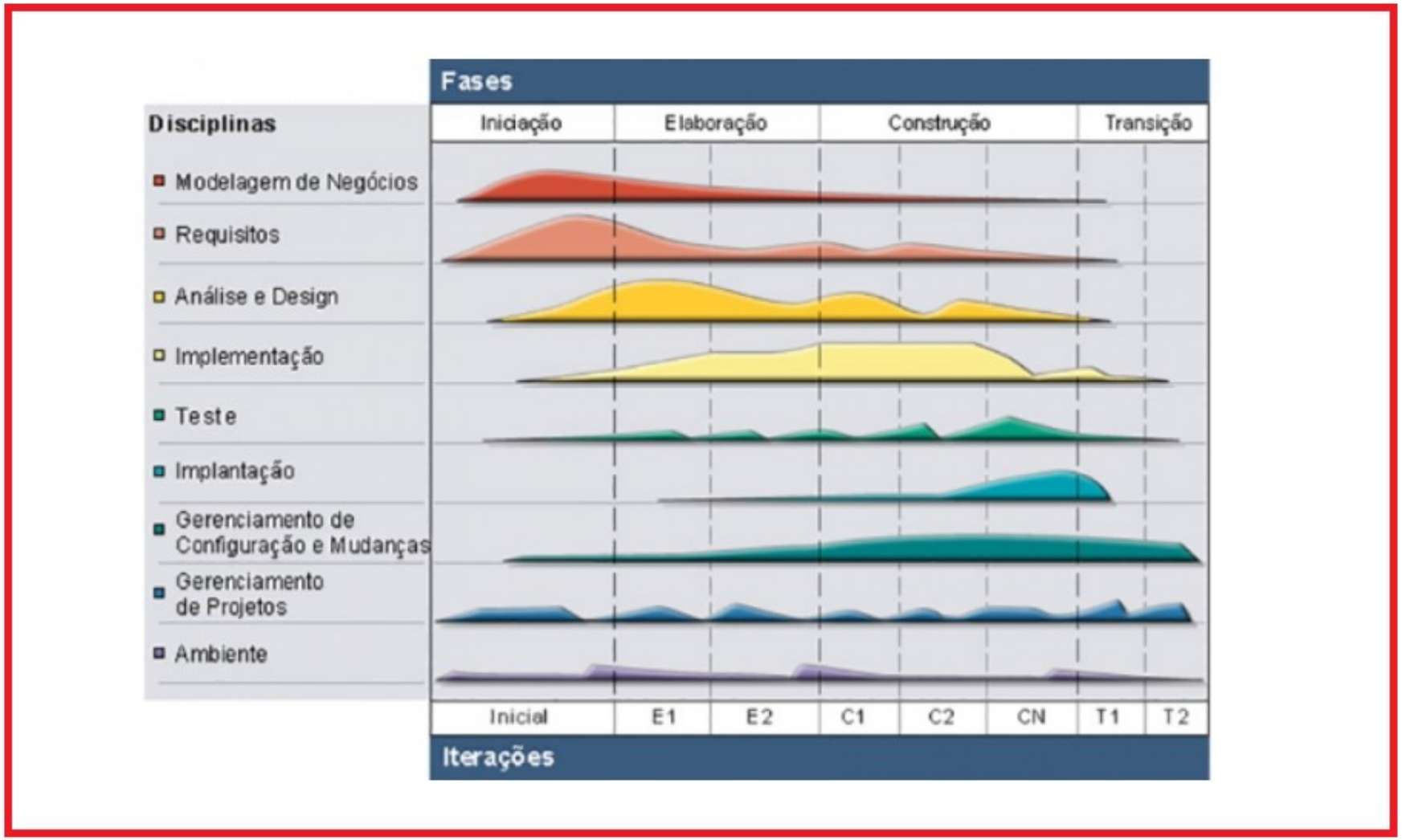
Exercício 5 - Solução

- Justificando sua resposta com base no tipo de sistema a ser desenvolvido, sugira o modelo de processo de software mais apropriado que poderia ser usado como base para gerenciar o desenvolvimento dos seguintes sistemas:
- a. um sistema para controlar um antibloqueador de freios de um automóvel;
  - Cascata porque os requisitos são bem definidos e pouco propensos a mudanças
- b. um sistema de realidade virtual para apoiar a manutenção de carros;
  - Desenvolvimento evolucionário, pois esse sistema é complexoe por isso precisa de constante reanálie dos requisitos
- c. Um sistema de contabilidade de uma universidade, que substituirá um já existente;
  - Engenharia de software baseada em componentes pois o sistema possui uma versão anterior que pode possuir um número considerável de componentes reusáveis.
- d. Um sistema interativo que permite aos passageiros encontrar os horários dos trens por meio de terminais instalados nas estações;
  - Desenvolvimento evolucionário, pois apesar de ser improvável que haverá alguma mudança nos requisitos,é muito provável que durante o processo de desenvolvimento apareçam vários novos.

Modelo RUP (Rational Unified Process)

- O Processo Unificado da Rational conhecido como RUP (Rational Unified Process), é um processo de engenharia de software criado para apoiar o desenvolvimento orientado a objetos, fornecendo uma forma sistemática para se obter vantagens no uso da UML.
- Foi criado pela Rational Software Corporation e adquirido em fevereiro de 2003 pela IBM.

Modelo RUP (Rational Unified Process)



Fases do RUP

- A iniciação, também conhecida como concepção, é a fase em que se estabelece o escopo do projeto de software, levantando-se uma visão geral do produto final. A iniciação tem como objetivo estabelecer o escopo, detalhar os casos de uso crítico do software, estimar o custo, os riscos e preparar o ambiente para o projeto. Se o projeto for de pouca importância ou inviável, ele pode ser cancelado após essa fase.
- Na elaboração, identificam-se os casos de uso principais e elaboram-se o sistema em iterações. A elaboração tem como objetivo garantir que a arquitetura, os requisitos e os planos estejam estáveis, que os riscos identificados sejam reduzidos, permitindo a criação de protótipos e estabelecer um ambiente de suporte.

Fases do RUP

- A construção é a etapa do desenvolvimento do software. Nela, produzem-se o código fonte do produto na linguagem de programação escolhida pela equipe do projeto. Os objetivos principais da construção compreendem reduzir os custos da implementação, obter a qualidade, concluir a análise, o design, a implementação e os testes das funcionalidades necessárias, desenvolver o produto de software, bem como, verificar se as funcionalidades foram finalizadas e se os usuários estão prontos para receber o sistema em ambiente de produção.
- A transição é a fase final do RUP. Nela, ocorre a validação e a entrega definitiva do software para o cliente. Essa etapa normalmente inclui a entrega do sistema, teste beta para validação das funcionalidades, conversão de bancos de dados operacionais, treinamento com os usuários, ajustes no sistema e a avaliação do produto.

Workflows do RUP

- Os workflows são definidos da seguinte maneira:
  - Modelagem de negócios: os processos de negócios são modelados com a utilização de casos de uso de negócio;
  - Requisitos: os casos de usos são criados para modelar os requisitos do software;
  - Análise e projeto: cria-se um modelo de projeto com base em modelos de arquitetura, de componente, de objeto e de sequência;
  - Implementação: os componentes do software são implementados;

Workflows do RUP

- Teste: o teste é um processo iterativo e é efetuado em conjunto com a implementação do sistema;
- Implantação: cria-se uma versão do produto e instala-a no local de trabalho dos usuários;
- Gerenciamento de configuração e mudanças: esse workflow serve como apoio à gerência do projeto em relação às mudanças no sistema;
- Gerenciamento de projetos: esse workflow de apoio gerencia o processo de desenvolvimento do software;
- Ambiente: este workflow relaciona-se à disponibilização de ferramentas de software adequadas para a equipe de desenvolvimento.

Escolha de um Modelo de Processo de Software

- Natureza do projeto e da aplicação
- Métodos e ferramentas a serem usados
- Controles e produtos que precisam ser entregues

Referências Bibliográficas

- PRESSMAN, Roger; MAXIM, Bruce. Engenharia de software: uma abordagem profissional. 8.ed. Bookman, 2016. E-book. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788580555349
- SOMMERVILLE, Ian. Engenharia de software. 9. ed. São Paulo: Pearson Prentice Hall, 2011. E-book. Disponível em: https://plataforma.bvirtual.com.br/Leitor/Publicacao/2613/epub/0
- Anotações de aula cedidas pelos professores: Jader Marques da Silva, Jean Paul Lopes, Osmar de Oliveira Braz Junior, Richard Henrique de Souza