
Desenvolvimento de Software Multiplataforma

Engenharia de Software 1

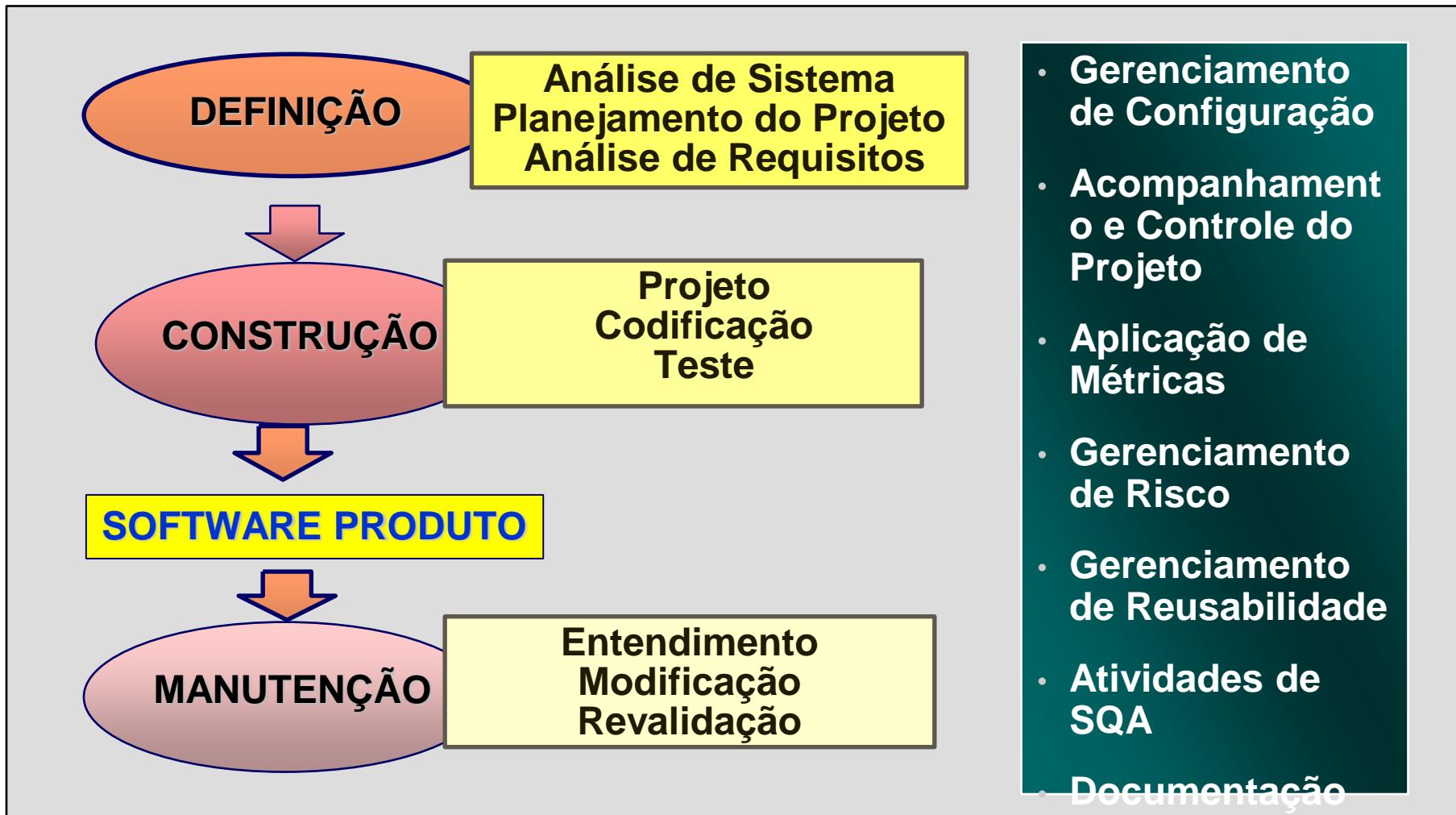
Prof. Cida Zem

aparecida.lopes01@fatec.sp.gov.br

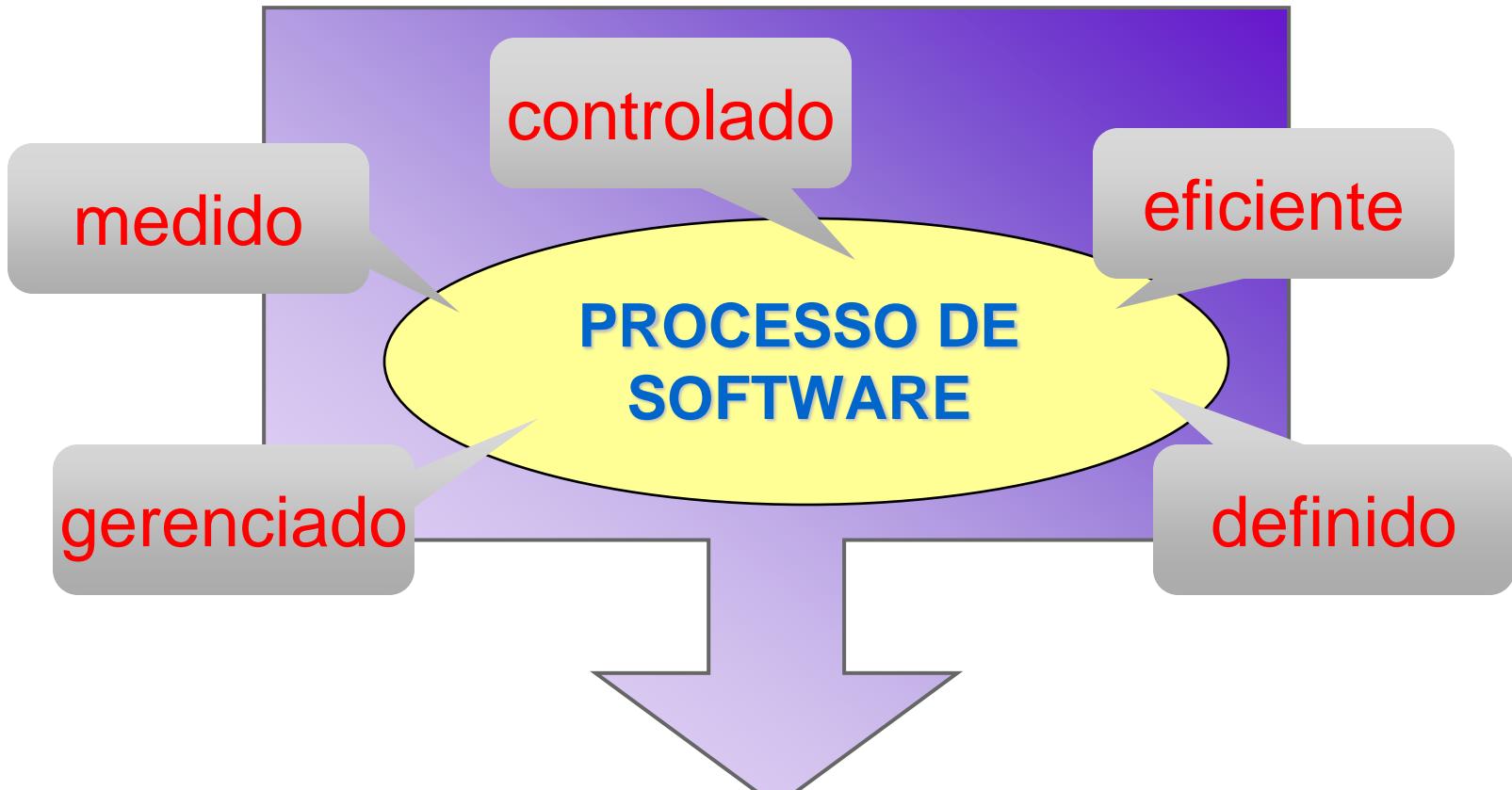
Modelos de Processo de Software

Processo de Software

Fases Genéricas



Um Processo de Software com Qualidade



Modelos de Processo de Software

Modelos de Processo de Software

- Existem vários modelos de processo de software.
 - **Paradigmas de Engenharia de Software.**
- Genéricos, abstrações do processo.
 - Usados para explicar diferentes abordagens para o desenvolvimento de software.
- Tentativa de colocar ordem em uma atividade inherentemente caótica.

Tópicos da Aula

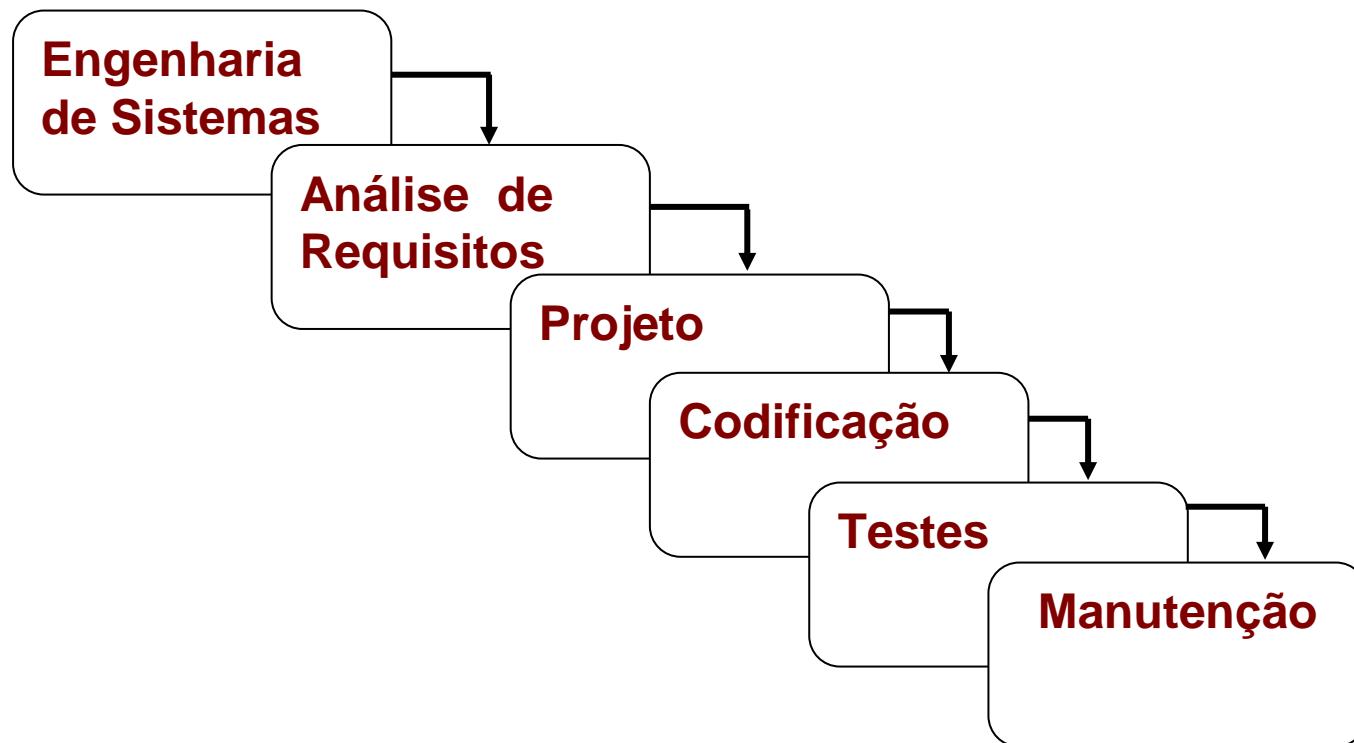
- Modelos de Processo de Software
 - Cascata
 - Modelo em V
 - RAD
 - Prototipação
 - Incremental
 - RUP
 - Espiral

O Modelo Cascata

- Modelo mais antigo e mais amplamente utilizado da Engenharia de Software.
 - **Paradigma do Ciclo de Vida Clássico.**
 - **Modelo Seqüencial Linear.**
- Sugere uma abordagem sistemática seqüencial para o desenvolvimento de software.
 - Tem início em nível de sistema e avança ao longo da análise, projeto, codificação, teste e manutenção.

O Modelo Cascata

► Modelado em função do ciclo convencional de engenharia. O resultado de uma fase constitui-se na entrada de outra.



O Modelo Cascata

Engenharia de Sistemas

- Estabelecimento de requisitos para **todos os elementos do sistema** e atribuição de um subconjunto desses requisitos para o software.
 - Visão essencial quando o software deve fazer interface com outros elementos (hardware, pessoas, banco de dados, etc.).
- Coleta de requisitos em nível do **sistema**, com uma pequena quantidade de **projeto** e **análise** de alto nível.

O Modelo Cascata

Análise de Requisitos de Software

- O processo de coleta dos requisitos é intensificado e concentrado especificamente no software.
- Para entender a natureza do programa a ser construído:
 - Deve-se compreender o domínio da informação do software, a função, o desempenho e a interface exigidos.
- Os requisitos (do sistema e do software) são documentados e revistos com o cliente.

O Modelo Cascata

Projeto

- Tradução dos requisitos do software para um conjunto de **representações** que podem ser avaliadas quanto à qualidade, antes que a codificação se inicie.
- Concentra-se em quatro atributos distintos do programa: *estrutura de dados, arquitetura do software, representações da interface e detalhes procedimentais*.
- Assim como os requisitos, o projeto deve ser documentado e torna-se parte da configuração de software.

O Modelo Cascata

Codificação (Geração de Código)

- Tradução das representações do projeto para uma linguagem de programação, resultando em instruções executáveis pelo computador.
 - Se o projeto for executado detalhadamente, a codificação pode ser executada mecanicamente.

O Modelo Cascata

Testes

- Concentram-se nos aspectos lógicos internos do software, garantindo que todas as instruções tenham sido testadas.
- Concentram-se nos aspectos funcionais externos a fim de descobrir erros e garantir que as entradas definidas produzam resultados reais que estejam em conformidade com os resultados esperados.

O Modelo Cascata

Manutenção

- Provavelmente o software deverá sofrer mudanças depois que for entregue ao cliente.
- Causas das mudanças:
 - Erros.
 - Adaptação do software para acomodar mudanças em seu ambiente externo.
 - Exigência do cliente para melhoramentos funcionais e de desempenho.
- Reaplica cada uma das fases precedentes a um programa existente.

Modelo Cascata: Problemas

- 💣 Projetos reais raramente seguem o fluxo seqüencial que o modelo propõe.
 - 💣 Modificações podem causar confusão à medida que o projeto prossegue.
- 💣 É difícil para o cliente estabelecer explicitamente todos os requisitos logo no início do projeto.
 - 💣 Dificuldade do modelo em acomodar a incerteza natural que existe no começo do projeto.

Modelo Cascata: Problemas

- 💣 O cliente deve ter paciência. Uma versão executável do software só fica disponível em uma etapa avançada do desenvolvimento.
 - 💣 Um erro grosseiro pode ser desastroso, se não for detectado até que o programa executável seja revisto.

Embora o **Modelo Cascata** tenha fragilidades, ele é significativamente melhor do que uma abordagem casual ao desenvolvimento de software.

O Modelo Cascata

- **Contribuições** importantes do modelo ao processo de desenvolvimento de software:
 - Produz um **padrão** no qual os métodos para análise, projeto, codificação, testes e manutenção podem ser situados.
 - Impõe **disciplina, planejamento e gerenciamento**.
 - A implementação do produto deve ser **postergada** até que os objetivos tenham sido completamente entendidos.

Modelo em V

O modelo em cascata é colocado em forma de "V".

▷ Do lado esquerdo do V:

- da análise de requisitos até o projeto,

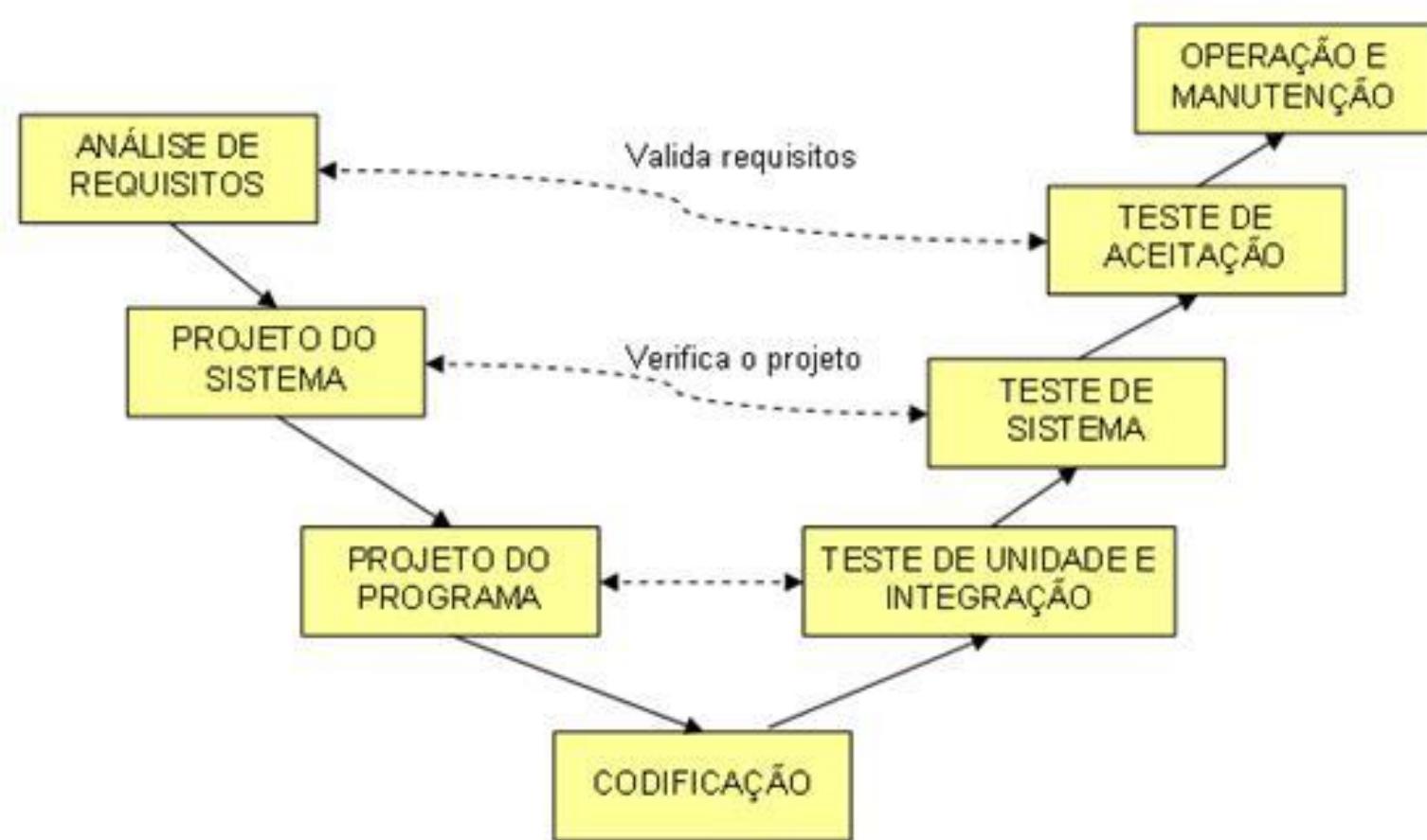
▷ No Vértice:

- codificação

▷ Do lado direito do V:

- testes, desenvolvimento, implantação e manutenção

Modelo em V



Modelo em V

Diferença do Cascata:

Ênfase à Verificação e Validação:

Cada fase do lado esquerdo gera um plano de teste
a ser executado no lado direito.

Modelo em V

Diferença do Cascata:

Ênfase à Verificação e Validação:

Cada fase do lado esquerdo gera um plano de teste
a ser executado no lado direito.

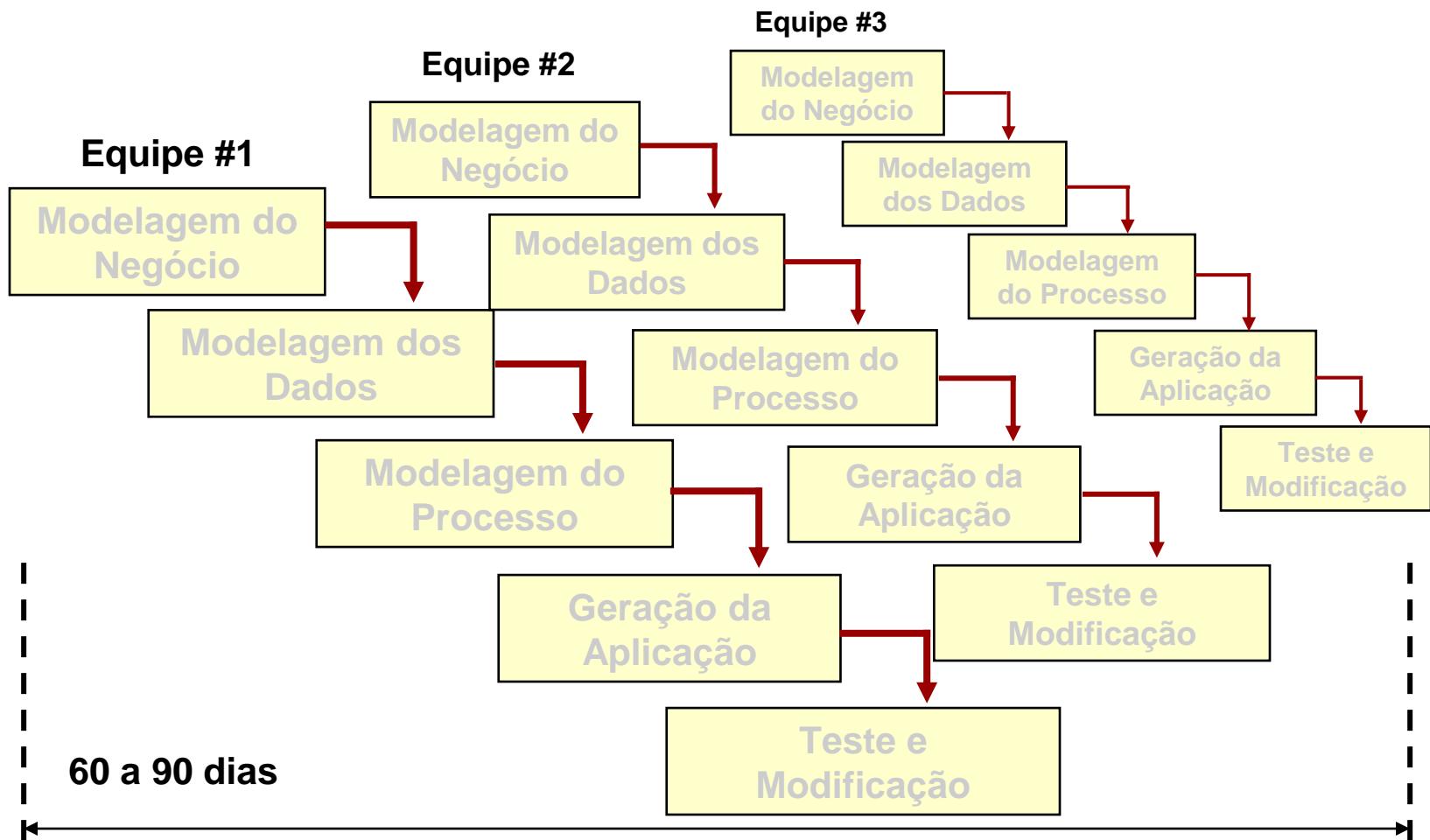
O Modelo RAD

- RAD (*Rapid Application Development*) é um modelo seqüencial linear que enfatiza um ciclo de desenvolvimento extremamente curto.
- Trata-se de uma adaptação de “alta velocidade” do **modelo seqüencial linear**.

O Modelo RAD

- ▷ A abordagem engloba as seguintes **fases**:
 1. Modelagem do Negócio
 2. Modelagem dos Dados
 3. Modelagem do Processo
 4. Geração da Aplicação
 5. Teste e Modificação

O Modelo RAD



O Modelo RAD

1- Modelagem do Negócio

- O fluxo de informação entre as **funções de negócio** é modelado de modo que responda as seguintes questões:
 - Que informação direciona o processo de negócio?
 - Que informação é gerada?
 - Quem a gera?
 - Para onde vai a informação?
 - Quem a processa?

O Modelo RAD

2- Modelagem dos Dados

- O fluxo de informação definido como parte da fase de modelagem dos negócios é refinado em um conjunto de **objetos de dados** necessários para apoiar os negócios.

- As **características** (atributos) de cada objeto são identificadas e o **relacionamento** entre esses objetos é definido.

O Modelo RAD

3- Modelagem do Processo

- Os objetos de dados definidos na fase de modelagem dos dados são transformados para obter o fluxo de informação necessário para implementar uma **função de negócio**.
- As descrições de processamento são criadas adicionando, modificando, excluindo ou recuperando objetos de dados.

O Modelo RAD

4- Geração da Aplicação

- *RAD* assume o uso de técnicas de 4^a geração (ferramentas automatizadas para facilitar a construção do software).
 - Delphi, Visual Basic, Asp.net, etc.
- O processo *RAD* trabalha para **reutilizar componentes** de programa existentes (quando possível) ou criar componentes reutilizáveis.

O Modelo RAD

5- Teste e Modificação

- Como o processo RAD enfatiza a reutilização, muitos dos componentes do programa já foram testados.
 - Isso **reduz o tempo de teste global**.
 - Entretanto... os novos componentes devem ser testados e todas as interfaces devem ser exaustivamente exercitadas.

O Modelo RAD

- Se uma aplicação de negócio puder ser **modularizada**, de modo que possibilite que cada função principal seja completada em um **curto prazo** (por ex: 30 a 90 dias), ela é candidata ao RAD.
 - Cada função principal pode ser tratada por uma equipe RAD distinta e depois integrada para formar o todo.

Modelo RAD: Problemas

- Para projetos grandes (mas escaláveis), RAD exige **recursos humanos** suficientes para criar um número adequado de equipes.
- Exige que desenvolvedores e clientes estejam comprometidos com **atividades continuamente rápidas**, necessárias para obter um **sistema completo** em um **período muito curto**.

Modelo RAD: Problemas

- Nem todos os tipos de aplicação são apropriadas para o RAD.
 - Deve ser possível a **modularização efetiva** da aplicação.
- Se o sistema não puder ser adequadamente modularizado, a construção dos componentes será problemática.

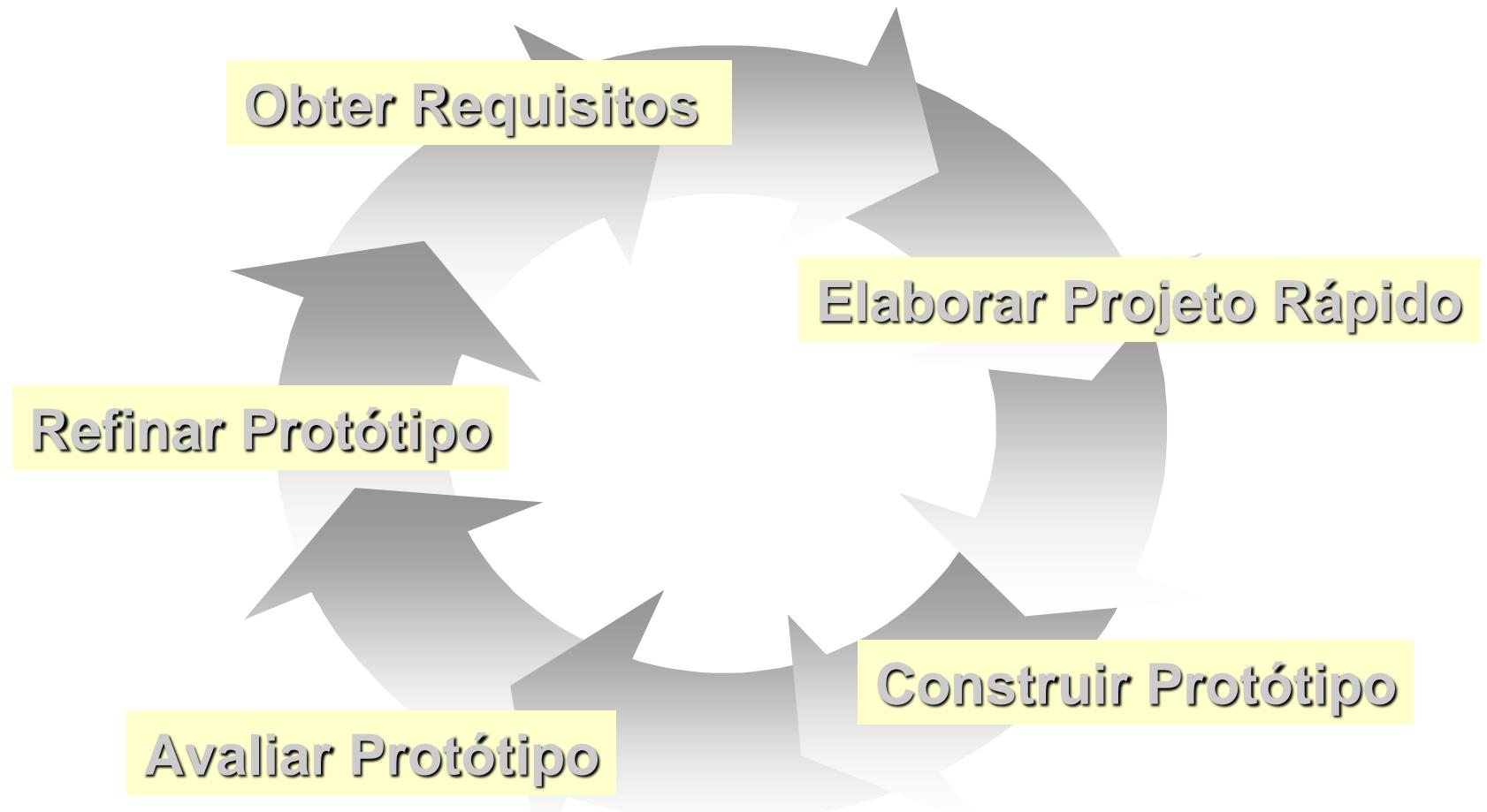
Modelo RAD: Problemas

- Quando **riscos técnicos** forem elevados, o RAD não é adequado.
 - A nova aplicação faz uso intenso de **novas tecnologias**.
 - Prototipação
 - O novo software exige um alto grau de **interoperabilidade** com programas existentes.

O Modelo de Prototipação

- Possibilita que o desenvolvedor crie um modelo (**protótipo**) do software que deve ser construído.
- O objetivo é entender os **requisitos do usuário** e, assim, obter uma melhor definição dos requisitos do sistema.
 - Apropriado para quando o cliente não definiu detalhadamente os requisitos.

O Modelo de Prototipação



O Modelo de Prototipação

Obtenção dos Requisitos

- Desenvolvedor e cliente:
 - Definem os objetivos gerais do software.
 - Identificam quais requisitos são conhecidos.
 - Identificam as áreas que necessitam de definições adicionais.

O Modelo de Prototipação

Projeto Rápido

- Representação dos aspectos do software que são visíveis ao usuário.
 - Abordagens de entrada.
 - Formatos de saída.

O Modelo de Prototipação

Construção do Protótipo

- Implementação rápida do protótipo.

O Modelo de Prototipação

Avaliação do Protótipo

- Cliente e desenvolvedor avaliam o protótipo.

O Modelo de Prototipação

Refinamento do Protótipo

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

O Modelo de Prototipação

Construção do Produto

- Identificados os requisitos, o protótipo deve ser **descartado** e a versão de produção deve ser construída considerando os **critérios de qualidade**.

Prototipação: Problemas

- O cliente não sabe que o software que ele vê **não** considerou, durante o desenvolvimento:
 - Qualidade global.
 - Manutenibilidade a longo prazo.
- O desenvolvedor freqüentemente faz uma **implementação comprometida**.
 - Utilizando o que está disponível com o objetivo de produzir rapidamente um protótipo.

O Modelo de Prototipação

- Ainda que possam ocorrer problemas, a prototipação é um ciclo de vida eficiente.
- *Definir as regras do jogo logo no começo.*
 - O cliente e o desenvolvedor devem concordar que o protótipo será construído para servir como um **mecanismo para definição dos requisitos**.
 - Deve ser **descartado** (ao menos em parte).
 - software real é submetido à engenharia visando à **qualidade** e à **manutenibilidade**.

Modelos Evolucionários (Evolutivos)

- 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 Evolucionários (Evolutivos)

- Requisitos de **produto** e de **negócio** mudam conforme o desenvolvimento prossegue.
- Requisitos básicos são bem conhecidos, mas os **detalhes** ainda devem ser definidos.
- **Prazos reduzidos** de mercado tornam impossível a conclusão de um produto completo.
 - Uma **versão reduzida** pode ser elaborada face à competitividade ou às pressões do negócio.

Modelos Evolucionários (Evolutivos)

- **Modelo Cascata:** desenvolvimento retilíneo.
 - O sistema completo estará pronto depois que a seqüência linear for concluída.
- **Prototipação:** entendimento dos requisitos.
 - Em geral, não é projetado para ajudar um sistema em produção.

A natureza evolutiva do software
não é considerada.

Modelos Evolucionários (Evolutivos)

- Modelos evolutivos são **iterativos**.
 - Possibilitam o desenvolvimento de **versões** cada vez mais completas do software.
- Modelo Incremental
- Modelo Espiral

O Modelo Incremental

- ▷ O Modelo Incremental combina:
 - Elementos do Modelo Cascata (aplicado repetidamente).
 - A filosofia iterativa da Prototipação.

O Modelo Incremental

- Objetivo: trabalhar junto ao cliente para descobrir seus requisitos, de maneira incremental, até que o produto final seja obtido.
 - A evolução acontece quando novas características são **adicionadas** à medida que são sugeridas pelo cliente.

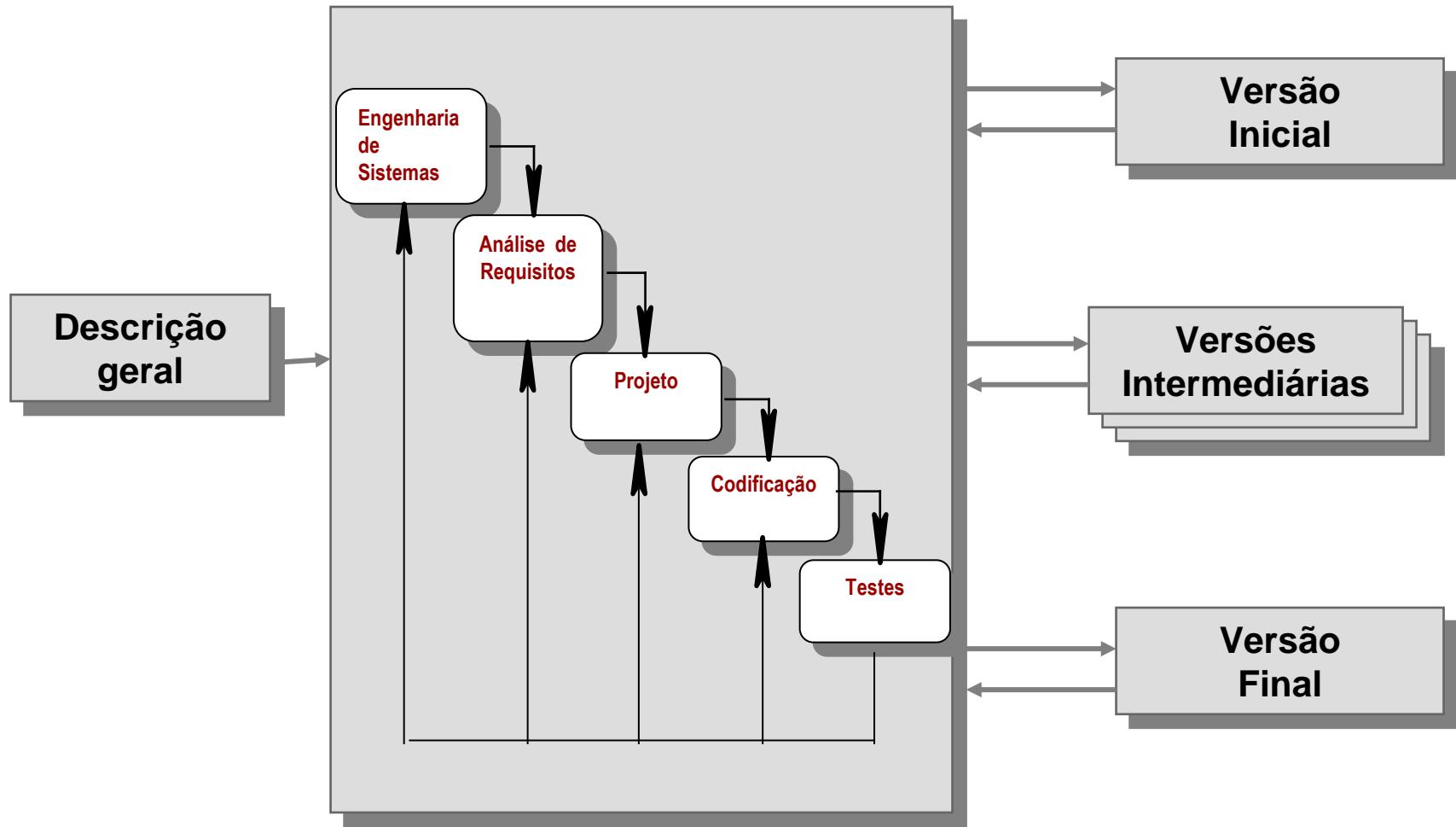
O Modelo Incremental

- Clientes identificam, em um esboço, as funções a serem fornecidas pelo sistema.
 - Funções **mais** e **menos** importantes.
 - Definem-se **estágios de entrega**.
 - Cada estágio fornece um **subconjunto** das funcionalidades do sistema.
 - **Incremento**.

O Modelo Incremental

- Os **incrementos** são elaborados utilizando-se o processo de desenvolvimento mais apropriado.
- Incrementos concluídos são **entregues** ao cliente e colocados em **operação**.

O Modelo Incremental



O Modelo Incremental

- A versão inicial é frequentemente o **núcleo** do produto (a parte mais importante).
- A funcionalidade do sistema **melhora** a cada novo estágio que é entregue.
 - Os incrementos são **versões simplificadas** do produto final.
- Oferecem **capacidades** que servem ao usuário.
- Constituem uma plataforma para **avaliação**.

O Modelo Incremental

- Importante quando é difícil estabelecer *a priori* uma especificação detalhada dos requisitos.
- Útil quando não há mão-de-obra disponível para uma implementação completa, dentro do prazo de entrega estabelecido.
- Os incrementos podem ser planejados de modo que os riscos técnicos possam ser gerenciados.

Modelo Incremental: Problemas

- Incrementos devem ser relativamente **pequenos** e produzir alguma **funcionalidade** para o sistema.
 - Difícil mapear os requisitos do cliente dentro de incrementos de tamanho correto.
- Difícil identificar **facilidades comuns**, exigidas por todos os incrementos.

Modelo RUP

- Derivado da UML e do Processo Unificado de Desenvolvimento de Software, o RUP, Rational Unified Process, é um modelo de processo iterativo e incremental, dividido em fases, orientado a casos de uso.
- Possui framework (esqueleto) de processo e manuais que guiam na utilização das melhores práticas de especificação de projeto.

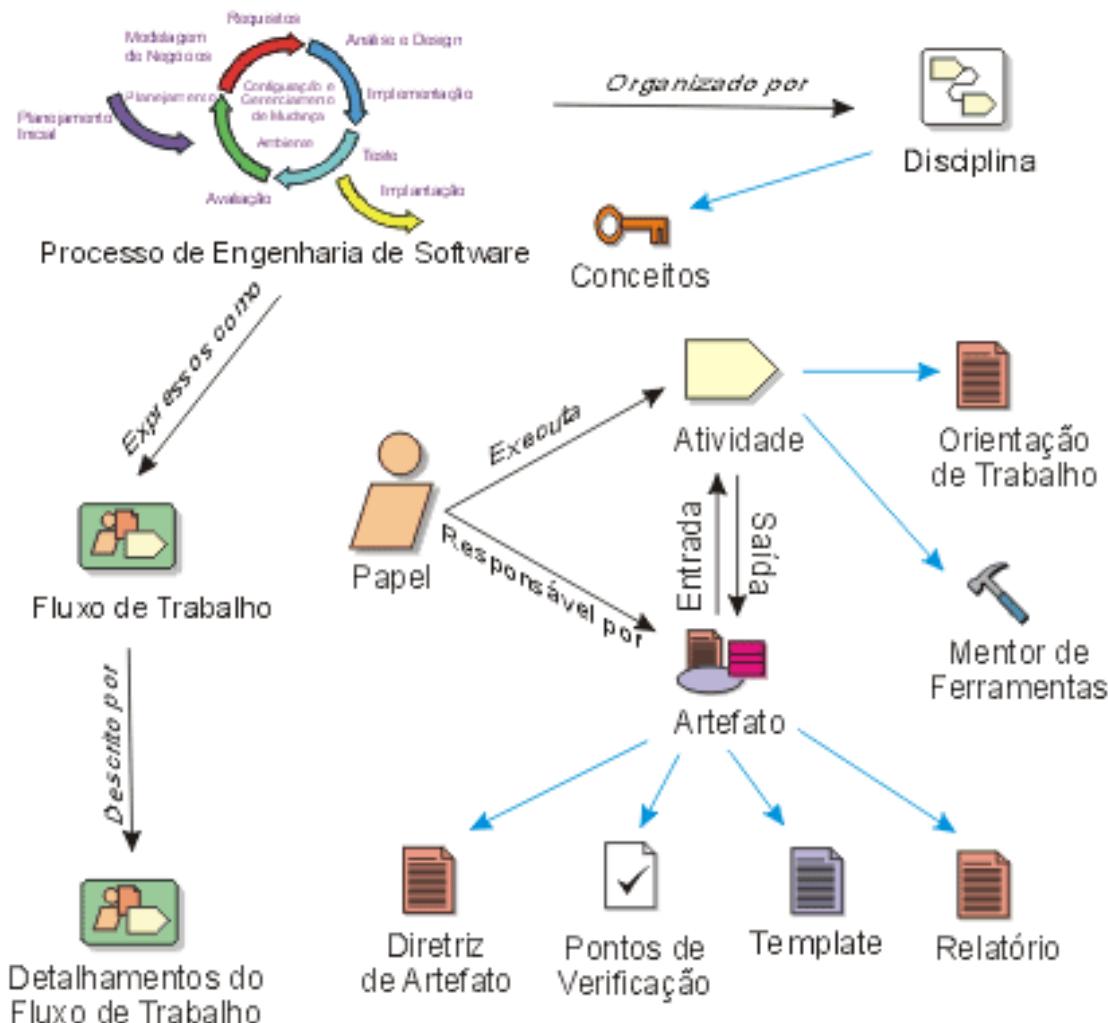
Modelo RUP

- O objetivo do RUP é produzir software com qualidade (melhores práticas de engenharia de software) que satisfaça as necessidades dos clientes dentro de um prazo e orçamento estabelecidos.
- Este modelo foi desenvolvido pela Rational Software Corporation e adquirido pela IBM.

Modelo RUP

- “IBM Rational Unified Process®, ou RUP, é uma plataforma de processo de desenvolvimento de software configurável que oferece melhores práticas comprovadas e uma arquitetura configurável

Modelo RUP



Modelo RUP

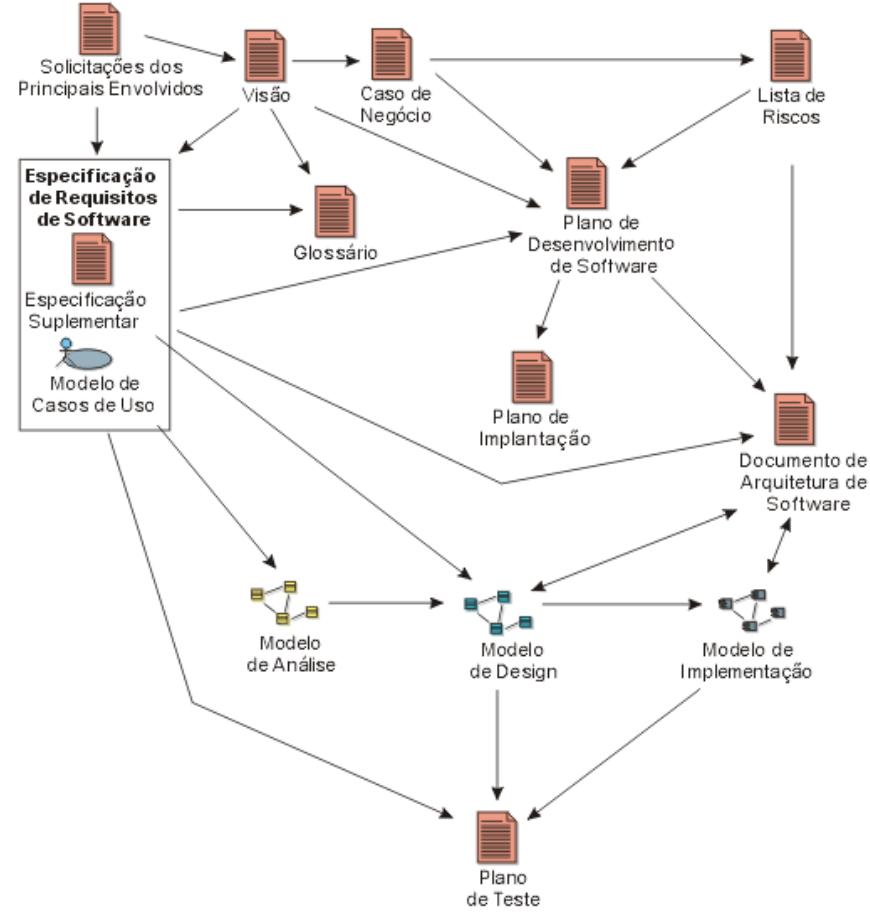
- O RUP possui quatro fases de negócio:
 1. Concepção: define o escopo do projeto, ou “business case”; onde é julgado se o projeto deve ir adiante ou ser cancelado.
 2. Elaboração: elabora modelo de requisitos, arquitetura do sistema, plano de desenvolvimento para o software e identificar os riscos.

Modelo RUP

3. Construção: constrói o software e a documentação associada.
4. Transição: finaliza produto, define-se plano de entrega e entrega a versão operacional documentada para o cliente.



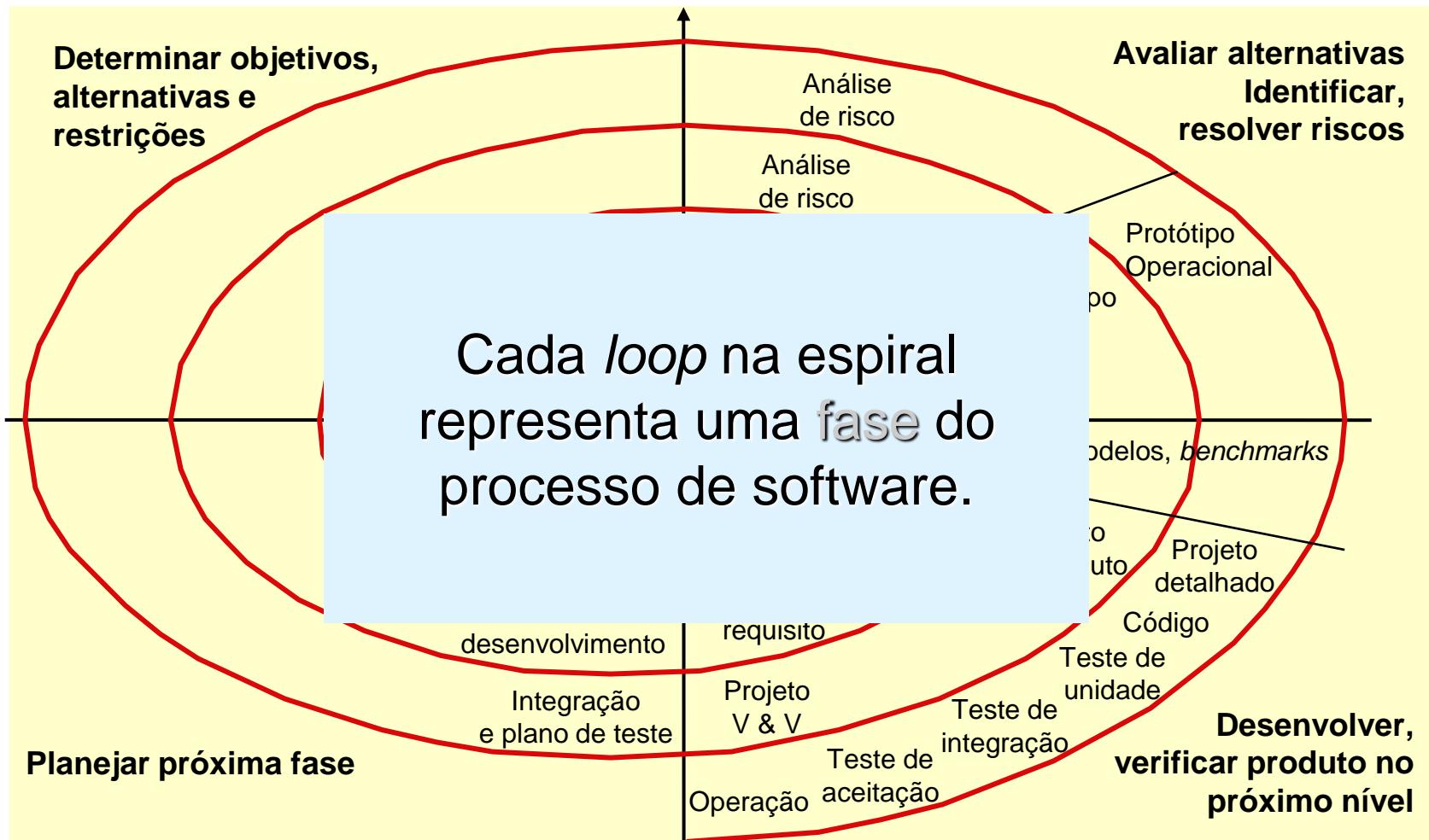
Modelo RUP



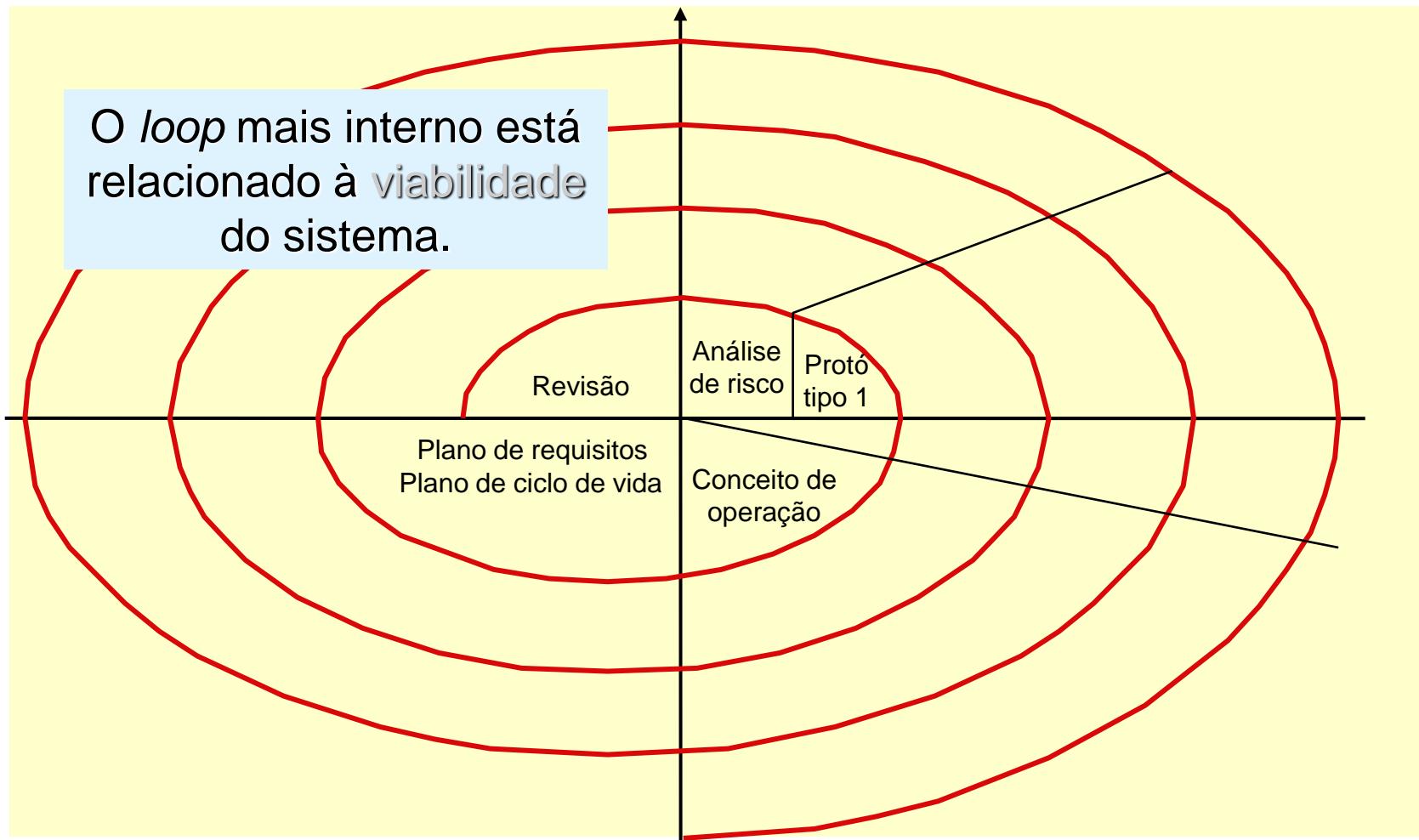
O Modelo Espiral

- O Modelo Espiral combina:
 - A natureza **iterativa** da Prototipação.
 - Os aspectos **controlados** e **sistemáticos** do Modelo Cascata.
- Fornece potencial para o desenvolvimento rápido de **versões incrementais** do software.
- Dividido em um certo número de atividades ou **regiões de tarefa**.
 - Existem, tipicamente, de 3 a 6 regiões de tarefa.

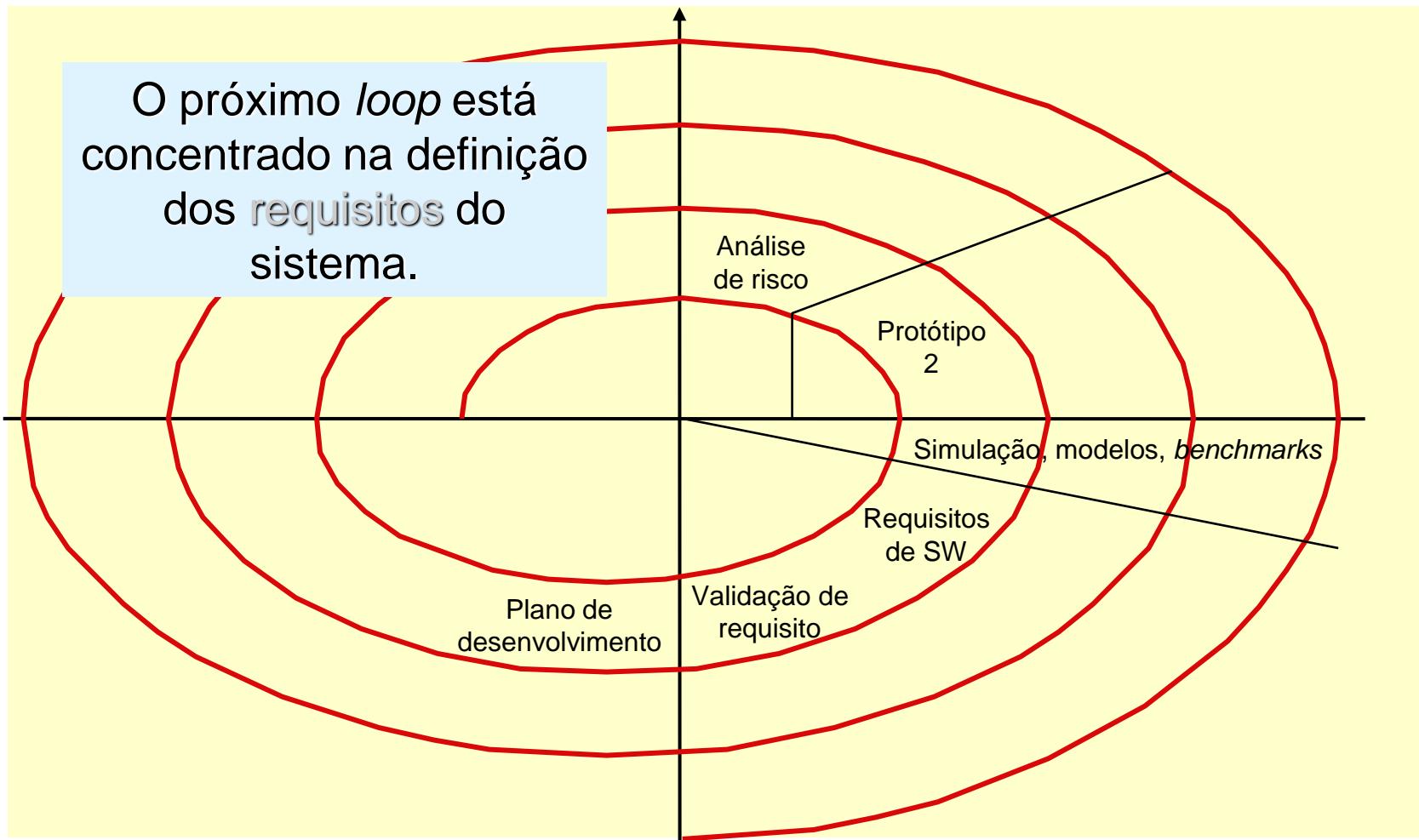
O Modelo Espiral



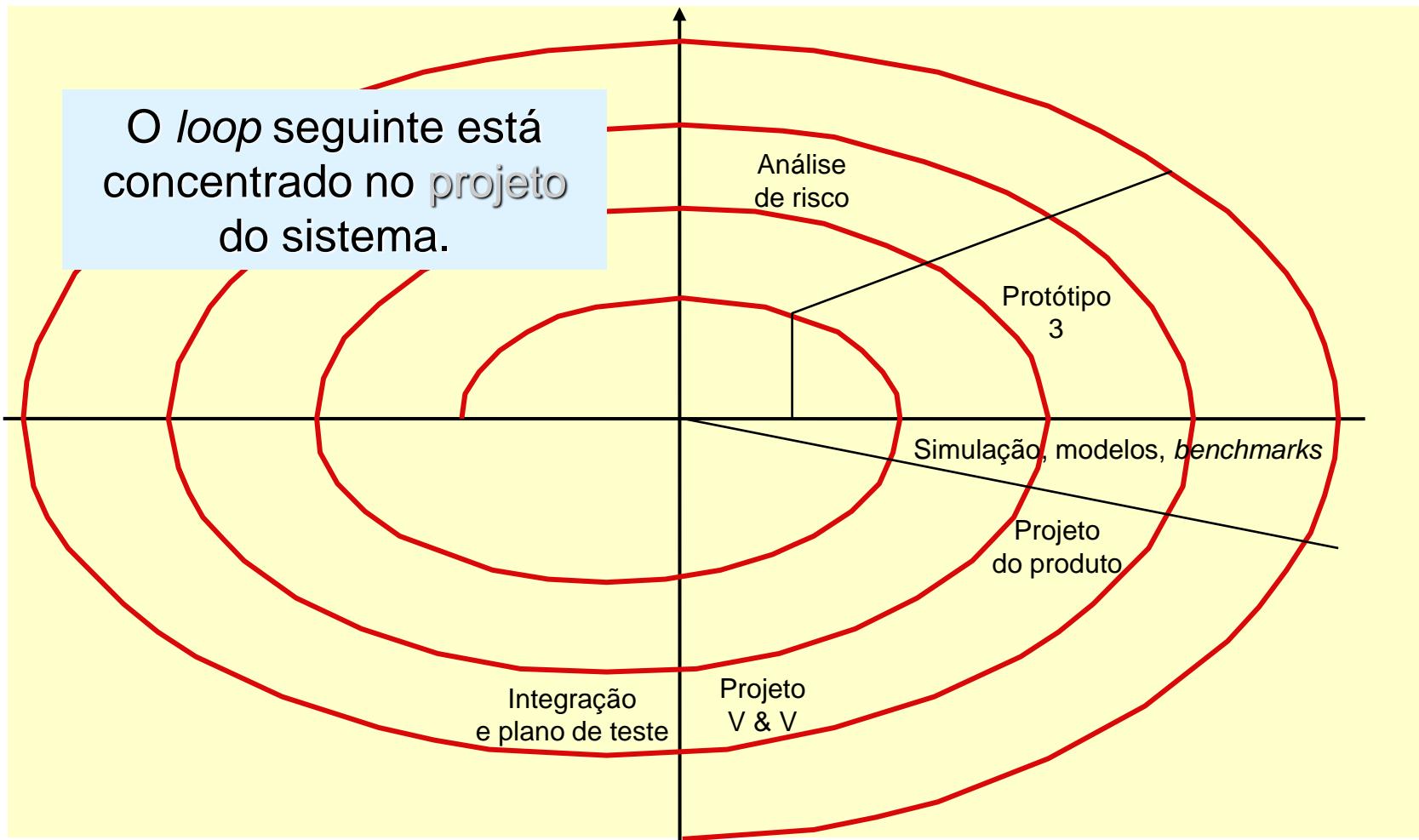
O Modelo Espiral



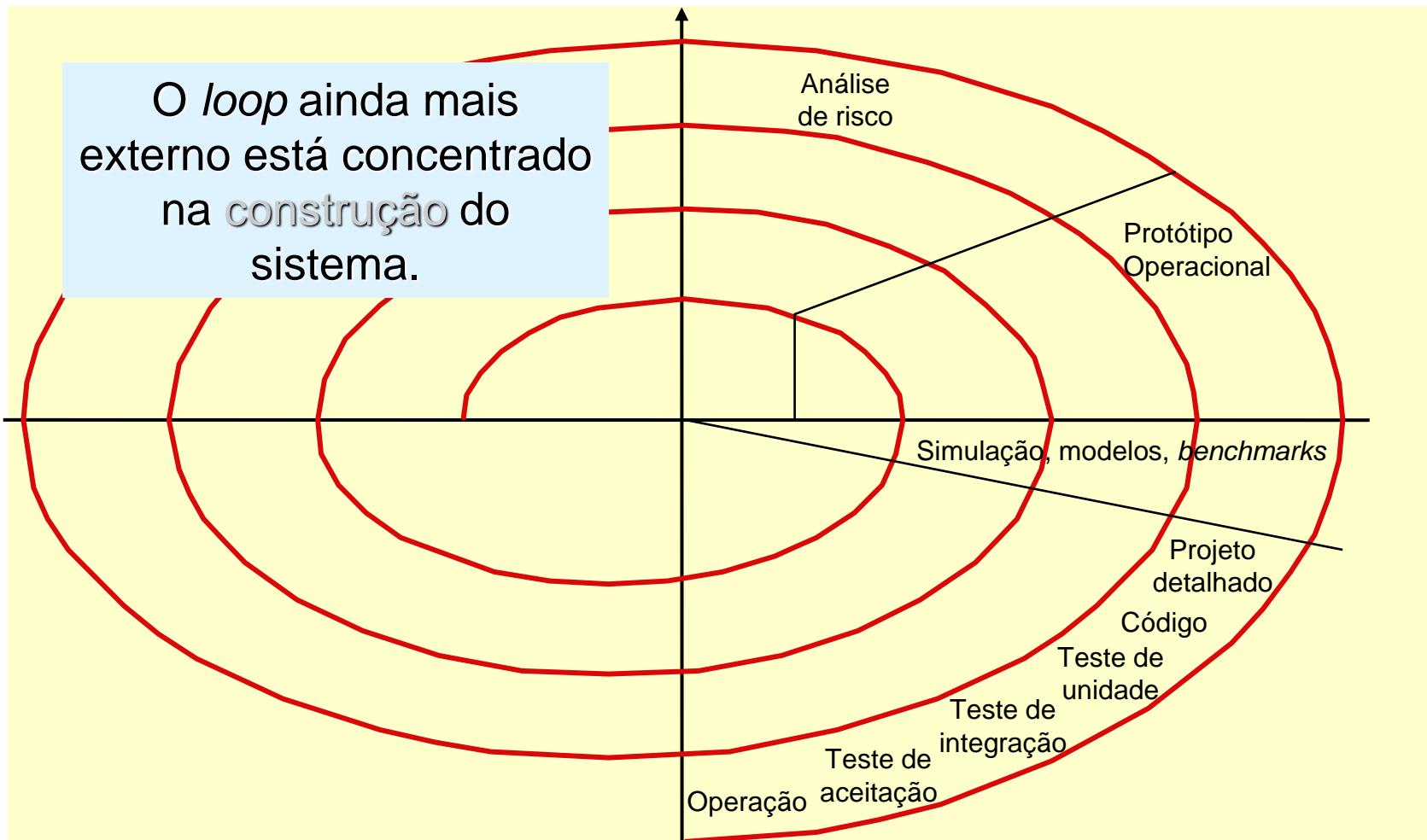
O Modelo Espiral



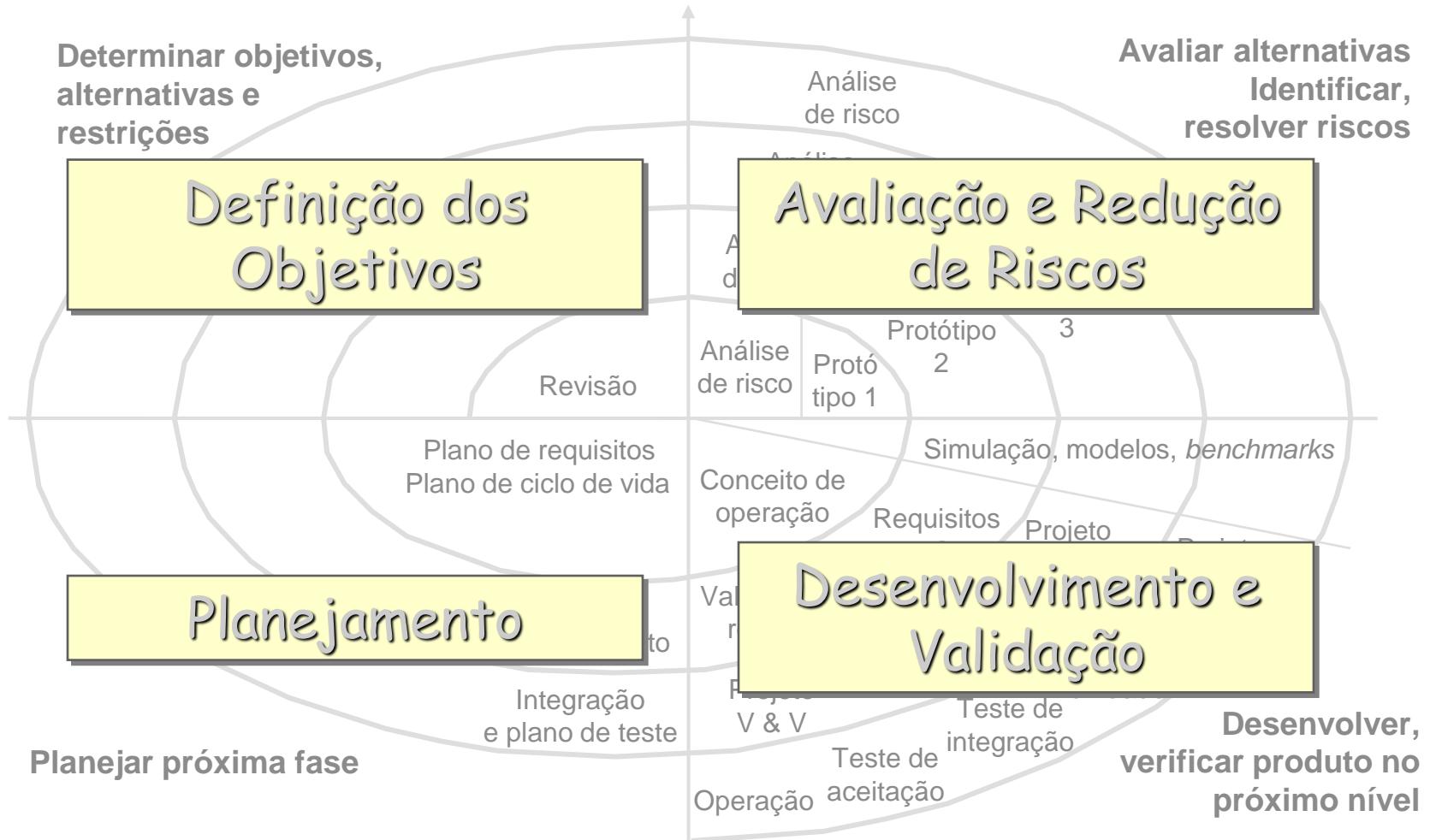
O Modelo Espiral



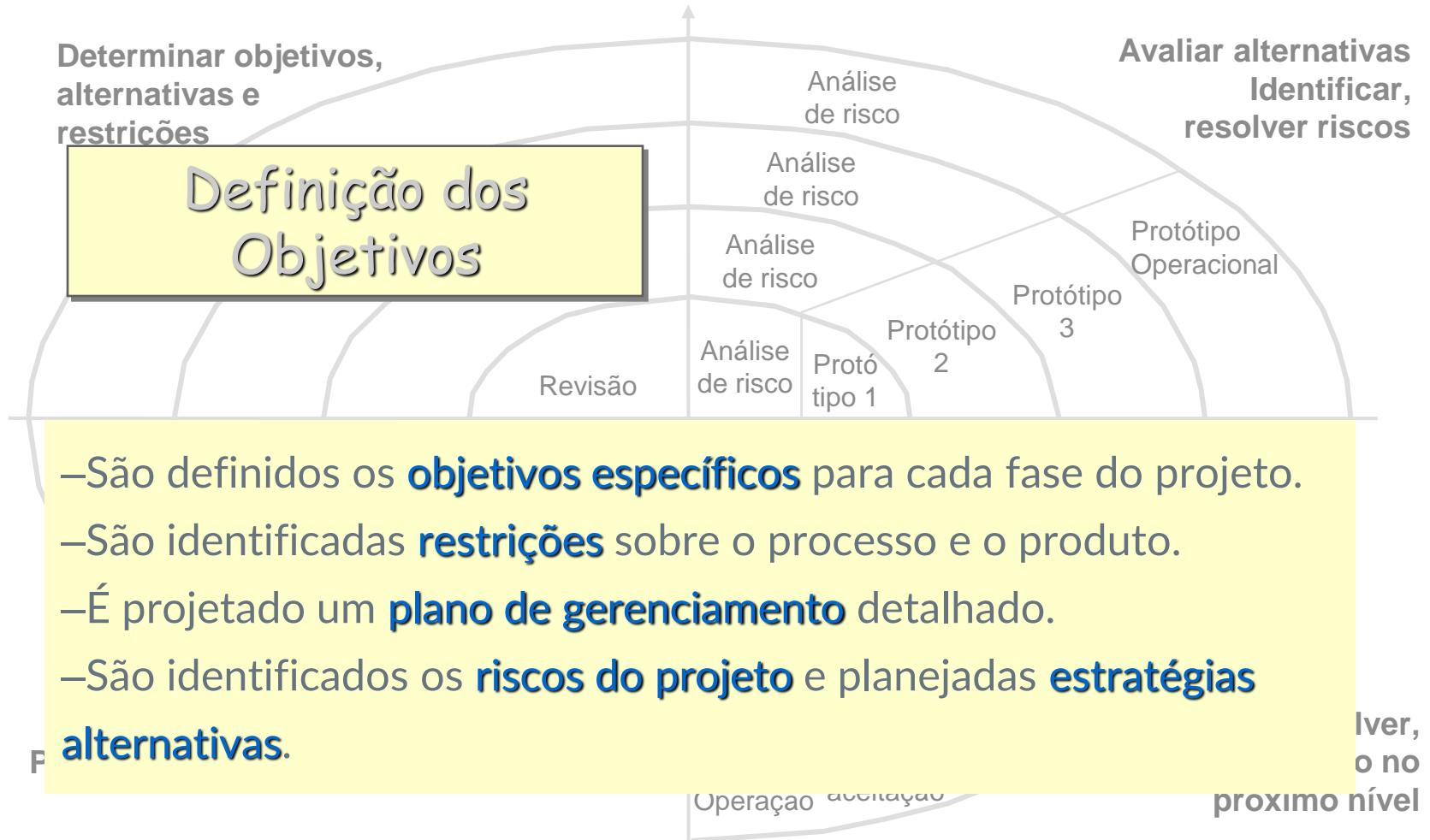
O Modelo Espiral



O Modelo Espiral



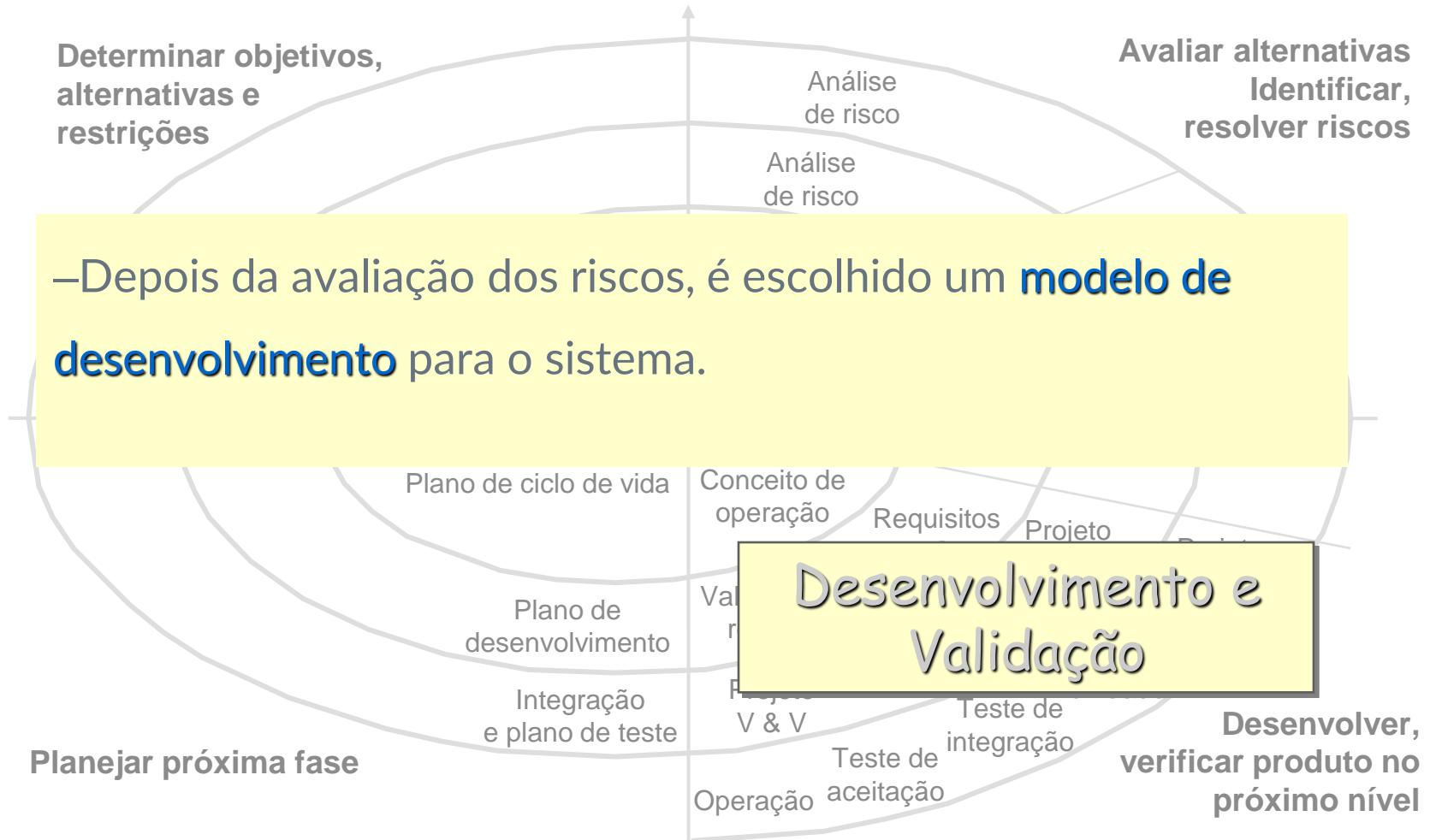
O Modelo Espiral



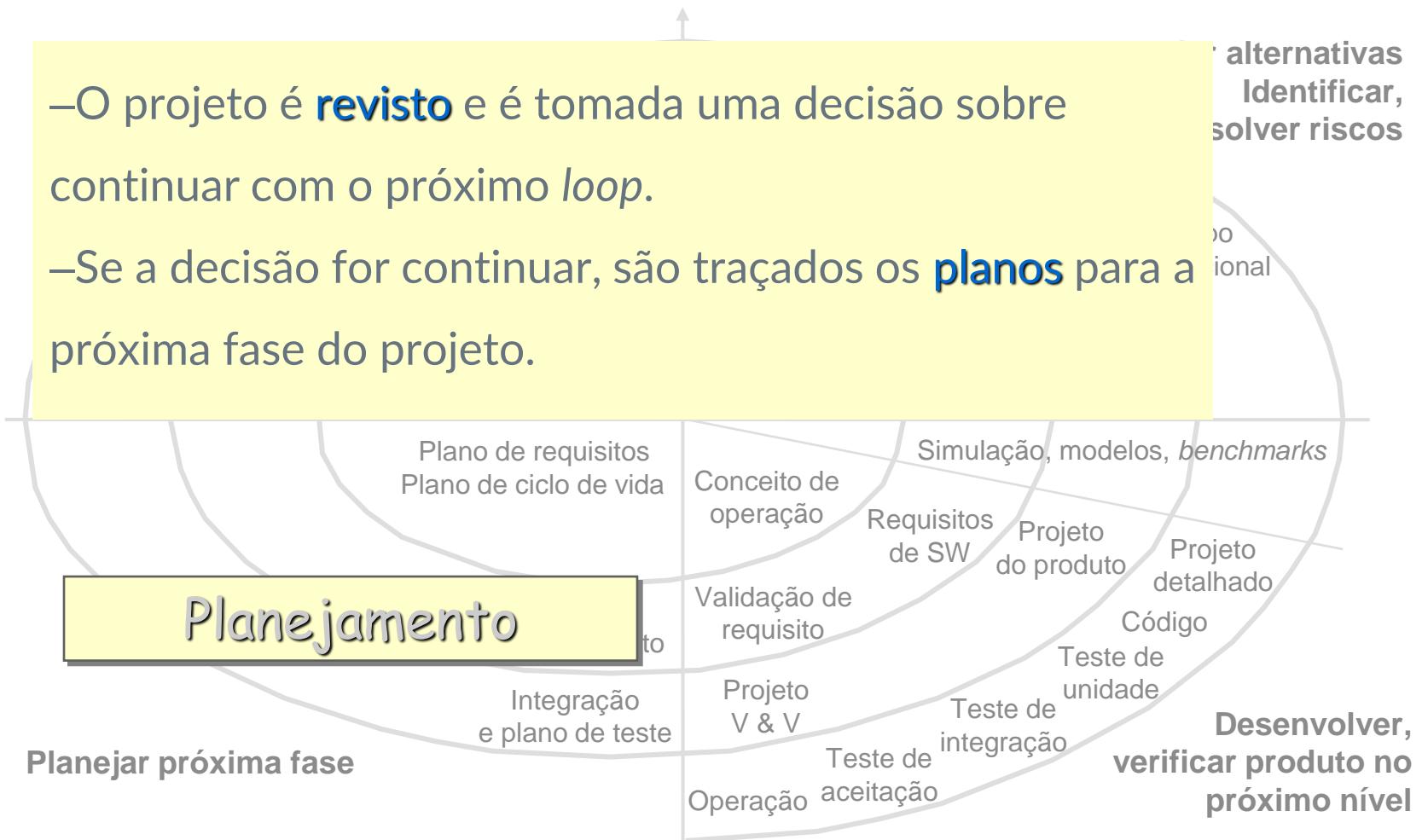
O Modelo Espiral



O Modelo Espiral



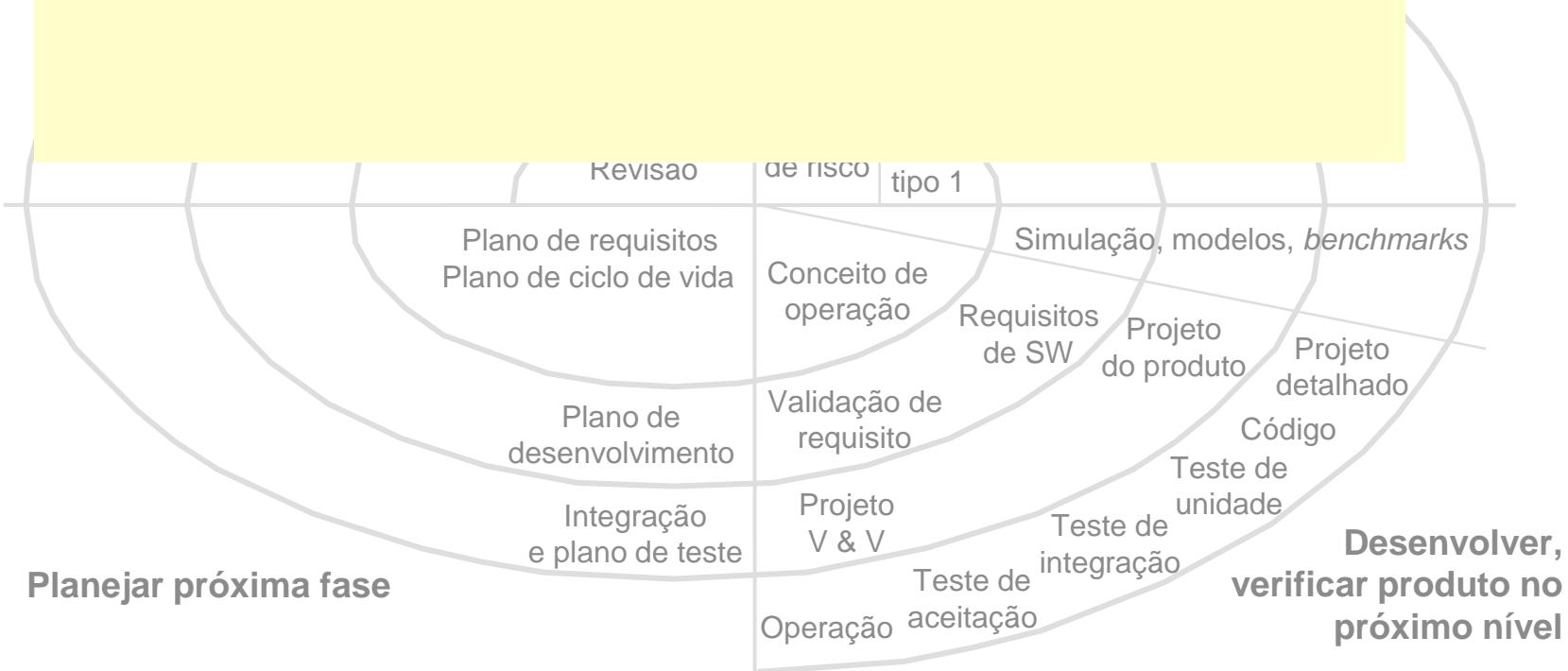
O Modelo Espiral



O Modelo Espiral

- Não existem **fases fixas** no modelo.
- A **gerência** decide como estruturar o projeto em fases.

alternativas
entificar,
er riscos



O Modelo Espiral

- Engloba as melhores características do Modelo Cascata e da Prototipação, adicionando um novo elemento: a **Análise de Riscos**.
 - Risco**: algo que pode dar errado.
 - Segue a abordagem de passos sistemáticos do Modelo Cascata, incorporando-os numa **estrutura iterativa**.
 - Usa a Prototipação em qualquer etapa da evolução do produto, como mecanismo de **redução** de riscos.

Modelo Espiral

- É, atualmente, a abordagem mais realística para o desenvolvimento de sistemas e software de **grande porte**.
- Exige a consideração direta dos **riscos técnicos** em todos os estágios do projeto.
 - Se aplicado adequadamente, pode reduzir os riscos antes que os mesmos fiquem problemáticos.

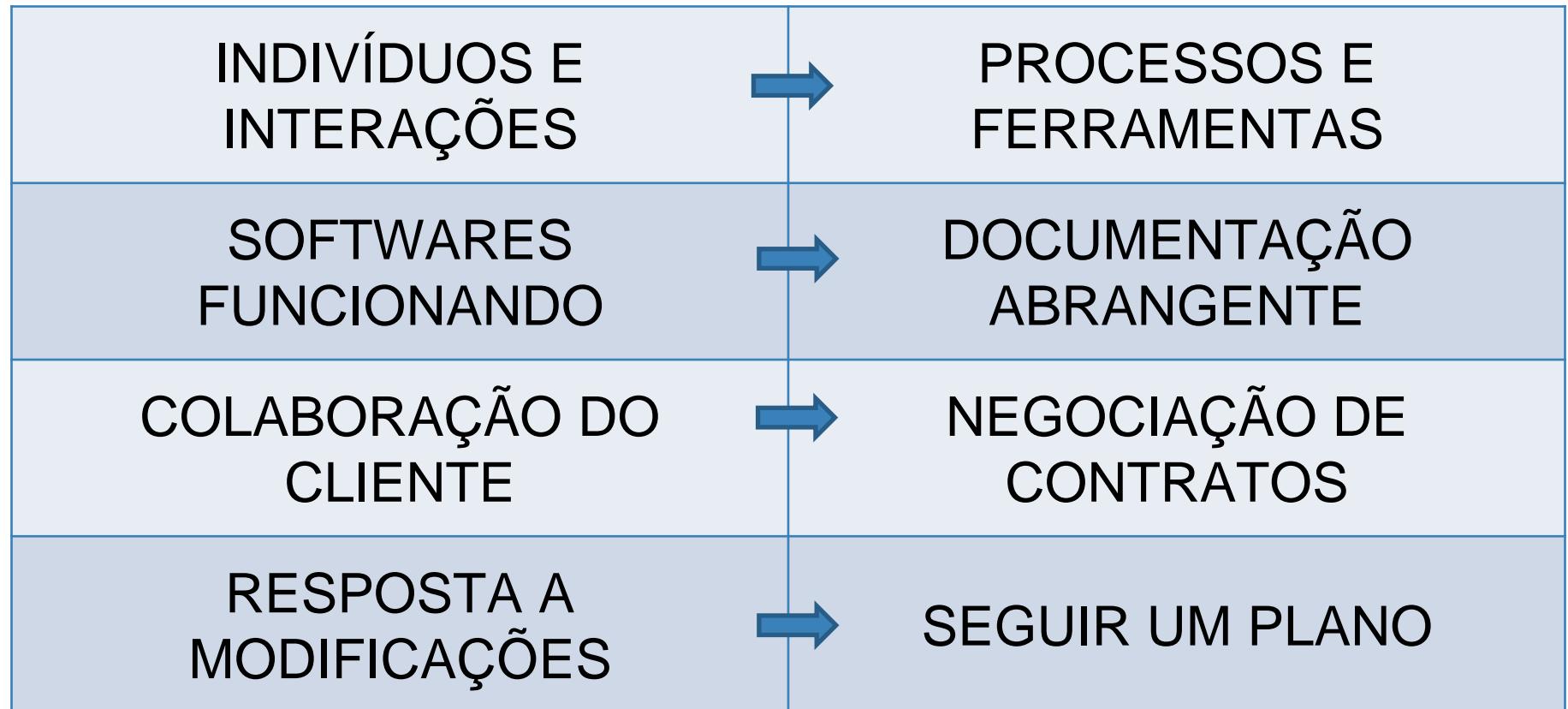
Modelo Espiral: Problemas

- Pode ser difícil **convencer** os clientes que uma abordagem evolutiva é controlável.
- Exige considerável **experiência** na determinação de riscos e depende dessa experiência para ter sucesso.

Quadro Comparativo entre os modelos

Modelo	Foco	Requisitos	1ª versão p/ cliente	Gerenciamento (1=mais simples)	Sistemas (tamanhocomplexidademáximos)
Cascata	Documento e artefato	Bem conhecido/congelado	Fim do ciclo	1	Simples
V	Planejamento de testes	Bem conhecido/congelado	Fim do ciclo	2	Simples
Incremental	Incrementos operacionais	Maior abstração / Tratado em módulos	Protótipos operacionais	3	Médio
Evolutivo	Evolução dos requisitos	Pouco conhecidos	Protótipos operacionais	4	Médio
RAD	Rapidez	Escopo restrito / Maior abstração / Tratado em módulos	Protótipos operacionais	4	Médio
Prototipagem	Dúvidas nos Requisitos	Abstratos	Protótipos não operacionais	5	Médio
Espiral	Análise de risco	Maior abstração / evoluídos com o tempo	Protótipos operacionais ou não operacionais	5	Complexos
RUP	Frameworks e boas práticas	Maior abstração / evoluídos com o tempo	Protótipos operacionais ou não operacionais	5	Complexos

Metodologias Ágeis



Concluindo...

- *Cascata, Prototipação, Evolutivos, Formais, Ágeis, ...*
- Para 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.
- Não são mutuamente exclusivos e freqüentemente são usados em conjunto.

Concluindo...

- Leiam:
- <http://blog.thiagofp.com.br/2010/08/02/sera-que-estamos-prontos-para-usar-metodologia-agil/>
- <http://zeluisbraga.wordpress.com/category/engenharia-de-software/metodos-ageis/>
- <http://www.dcc.ufla.br/infocomp/artigos/v3.2/art02.pdf>
- http://ccsl.ime.usp.br/agilcoop/casos_de_sucesso
- <http://br.blog.trello.com/scrum-metodologia-agil/>
- <http://www.mindmaster.com.br/scrum/>
- http://www2.comp.ufscar.br/~bruno_abrahao/Artigos/Crystal.pdf